PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Sur l'utilisation de réseaux de neurones dans un système de recommandations réciproques
Auteur: Author:	Antoine Lefebvre-Brossard
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lefebvre-Brossard, A. (2018). Sur l'utilisation de réseaux de neurones dans un système de recommandations réciproques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3262/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/3262/
Directeurs de recherche: Advisors:	Michel Desmarais
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

SUR L'UTILISATION DE RÉSEAUX DE NEURONES DANS UN SYSTÈME DE RECOMMANDATIONS RÉCIPROQUES

ANTOINE LEFEBVRE-BROSSARD
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOÛT 2018

© Antoine Lefebvre-Brossard, 2018.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

SUR L'UTILISATION DE RÉSEAUX DE NEURONES DANS UN SYSTÈME DE RECOMMANDATIONS RÉCIPROQUES

présenté par : LEFEBVRE-BROSSARD Antoine
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. PAL Christopher J., Ph. D., président
M. DESMARAIS Michel C., Ph. D., membre et directeur de recherche
M. CHENG Jinghui, Ph. D., membre

DÉDICACE

À tous ceux qui n'ont pas eu les mêmes opportunités.

REMERCIEMENTS

Je souhaite d'abord remercier mon directeur de recherche Michel Desmarais qui m'a supporté tout au long de ma trop longue maîtrise en me procurant encouragements, aide et opportunités de grandir et d'apprendre par des charges de laboratoires et de cours.

Mes remerciements vont aussi à mes collègues et amis qui m'ont aidé à organiser mes idées de recherche en en discutant avec moi ou en me changeant les idées.

Je tiens aussi à remercier ma famille qui m'ont supporté tout en ne sachant pas tout à fait ce sur quoi je travaillais.

Finalement, je remercie E180 pour avoir fourni les données ainsi qu'une partie des fonds ayant servi à financer un stage. Je remercie l'organisme Mitacs pour avoir fourni l'autre partie de ces fonds.

RÉSUMÉ

Le domaine de la recommandation de personne à personne comporte de multiples applications, mais il demeure moins bien étudié que celui de la recommandation de produits ou de services. Contrairement à la recommandation d'items, la recommandation de personnes doit tenir compte de la possibilité que la recommandation ne plaise pas nécessairement dans les deux sens, ce qui impose des difficultés supplémentaires et augmente la complexité.

Dans les dernières années, les algorithmes à base de réseaux neuronaux ont su tirer parti des complexités présentes dans des domaines aussi divers que la vision informatique, le traitement du langage naturel et de la parole, et la génération d'images.

Bien que leur utilisation pour les systèmes de recommandations en général ait été étudiée, l'utilisation des réseaux neuronaux est encore peu ou pas explorée dans le domaine de la recommandation de personne à personne. Nous explorons cette avenue. Nous avons utilisé des données provenant d'une plateforme qui permet de connecter deux personnes voulant apprendre l'une de l'autre. Cette base de données possède à la fois des données implicites sous la forme de vues de profils, de messages ou de rencontres, et des données explicites, les descriptions des offres et demandes ainsi que des mots-clés.

L'algorithme créé pour faire les recommandations avec ces données est une combinaison de réseaux de neurones récurrents. Ceux-ci sont entraînés en essayant de prédire les données implicites à partir des données explicites. L'algorithme est comparé à l'approche classique d'analyse sémantique latente (*Latent Semantic Analysis*) sur la base des mesures de précision, rappel et score F1. Les résultats montrent que le nouvel algorithme prédit moins bien les données historiques lorsque peu de prédictions sont faites, mais que la qualité de celles-ci augmente plus vite avec le nombre de prédictions que le modèle comparé. Ceci se traduit par une meilleure précision lorsque le rappel est grand. Ce résultat est similaire lorsque les deux modèles sont augmentés d'une approche par filtres collaboratifs, bien que la différence s'amoindrit. L'utilisation d'agrégation des expertises par maximum ou moyenne ne semble pas avoir beaucoup d'effet pour l'un ou l'autre des modèles.

Ce mémoire introduit une nouvelle approche pour les systèmes de recommandations permettant d'entraîner des modèles utilisant des données dépendant uniquement de l'utilisateur pour faire des recommandations aussi complexes et diversifiées que celles faites par les filtres collaboratifs.

ABSTRACT

People-to-people recommandations is a relatively new domain of study compared to recommender systems in general. Contrary to recommender systems where items are recommended to people, the recommendation of people has to take into account the fact that recommendations may be not be as good in the reversed direction. This factor increases the complexity of the recommendation.

During the last few years, algortithms based on neural networks have been able to find patterns in domains as diverse as computer vision, natural language and speech processing, and image generation. While their use in recommender systems in general has been studied, work on this topic is still in its infancy and we find no contribution yet for people-to-people recommendation. We explore this avenue and use data from a platform where two people wanting to learn from another can connect. This dataset contains implicit data in the form of profile views, messages and meetings, as well as explicit data under the format of demand and offer textual descriptions and tags.

The people-to-people recommender system developed is a neural network based on recurrent neurons and trained by trying to predict the implicit data from the explicit data. It is compared to the classical Latent Semantic Analysis approach based on precision, recall and F1 score metrics. The results show that the new algorithm does not do as well a job predicting historical data when few predictions are made, but that the quality goes up more quickly with the number of predictions made than the compared model. This is shown by a better precision when recall is large. The result is similar when both models are augmented with collaborative filters, but with a smaller difference between them. The use of different pooling methods by maximum or mean doesn't seem to have much of an effect on either model.

This model introduces a new approach for recommender systems that enables them to use data depending only on the user to make recommendations as complex and diversified as those made by collaborative filters.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

NOMENCLATURE, LISTE DES SIGLES ET ABRÉVIATIONS

a	Un scalaire
$\boldsymbol{a}$	Un vecteur
a_i	Le $i$ème élément du vecteur $\boldsymbol{a}$
$\boldsymbol{A}$	Une matrice
A_{ij}	L'élément de la $i$ème ligne et $j$ème colonne de la matrice $\boldsymbol{A}$
$\boldsymbol{A_{i\cdot}}$	Le vecteur correspondant à la $i$ème ligne de la matrice $\boldsymbol{A}$
$\boldsymbol{A_{\cdot j}}$	Le vecteur correspondant à la $j$ème colonne de la matrice $\boldsymbol{A}$
$\mathbb{1}_{\{c\}}$	La variable indicatrice prenant la valeur 1 si c est vraie et 0 si c est fausse
RNN	Réseau de neurones récurrents (*Recurrent Neural Network*)
MLP	Perceptron multicouche (*Multilayer Perceptron*)

CHAPITRE 1 INTRODUCTION

Dans les dernières décennies, la quantité croissante de produits et services offerts et la possibilité de collecter des quantités importantes de données a nécessité de nouvelles techniques pour en faire le tri. Plutôt que de devoir filtrer manuellement ces quantités importantes d'offres changeant au fil du temps, des systèmes de recommandations ont été créés pour en présenter un petit nombre à l'utilisateur, personnalisé pour que chacune de ces recommandations soit appropriée et intéressante pour celui-ci. Ces systèmes ont permis le succès de plusieurs entreprises dont par exemple Amazon et Netflix.

Les dernières années ont aussi vu le succès d'algorithmes à base de réseaux neuronaux dans divers domaines, dont la vision informatique, la reconnaissance et génération de la parole et le traitement du langage naturel. Ceux-ci ont permis de créer des représentations plus complexes et complètes en profitant d'importantes quantités de données. L'apparition récente de systèmes de recommandations à partir de ces réseaux neuronaux a montré que ceux-ci peuvent obtenir des résultats intéressants et ainsi améliorer la qualité des recommandations offertes aux utilisateurs.

1.1 Définitions et concepts de base

Un système de recommandation est un programme informatique permettant de filtrer automatiquement un grand nombre d'éléments de façon à seulement garder ceux jugés les plus pertinents par l'algorithme. En général, ceux-ci peuvent filtrer une liste de produits, services ou personnes pour un utilisateur donné, mais on parlera par la suite d'*item* pour désigner de façon générale tout élément pouvant être proposé à un utilisateur. Trois types de systèmes de recommandations existent dans la littérature : les filtres collaboratifs, les systèmes à base de contenu (*content-based*) et les systèmes à base de connaissances (*knowledge-based*) (Aggarwal et al., 2016).

Les *systèmes de filtres collaboratifs* sont basés sur le comportement des utilisateurs. Un item sera jugé similaire à un autre si les utilisateurs les ont traités de façon similaire et de la même façon, deux utilisateurs seront similaires s'ils ont traité interagi avec les mêmes items de la même façon. Par exemple, dans un contexte de recommandation de films où les utilisateurs peuvent donner une appréciation sur ceux qu'il a vus, deux films seront considérés similaires si les utilisateurs leur ont donné une appréciation similaire, et deux utilisateurs seront similaires s'ils ont noté les mêmes films de façon identique. Ces systèmes de recommandations

peuvent aussi utiliser le comportement implicite des utilisateurs. Deux sites visités par des utilisateurs provenant de sites similaires seront considérés comme similaires. Deux classes de filtres collaboratifs existent, utilisateur-utilisateur et item-item, dépendamment de comment les similarités sont calculées.

Les *systèmes à base de contenu* utilisent, quant à eux, directement les données rattachées aux items. Par exemple, dans le même cas de recommandation de films, le texte du résumé du film peut être utilisé ou la présence d'acteurs, scénaristes ou directeurs communs. Ceux-ci n'ont pas besoin d'un historique de données pour faire les premières recommandations et ont donc un avantage sur les systèmes de filtres collaboratifs qui souffrent de cette problématique de début d'utilisation, nommé *cold-start problem* dans la littérature.

Finalement, les *systèmes à base de connaissances* utilisent des spécifications données par l'utilisateur, de l'information sur les items et une connaissance du domaine pour évaluer les items à recommander. Contrairement aux deux premières approches, dans celle-ci l'utilisateur joue un rôle direct dans la recommandation qui lui est offerte. Dans l'exemple de la recommandation de films, l'utilisateur pourrait spécifier un genre et seulement des films similaires lui seront proposés. Le désavantage de cette approche est qu'elle nécessite de créer la base de connaissances permettant de faire les recommandations et n'est donc pas aussi automatique que les approches précédentes.

L'autre domaine traité dans ce document est celui des *réseaux neuronaux*. Les réseaux neuronaux sont une famille d'algorithmes permettant de découvrir des structures dans un espace de données dans le but de prédire ou d'organiser celles-ci. Ceux-ci utilisent des transformations linéaires et des fonctions non-linéaires pour créer des hiérachies de représentations permettant d'identifier des structures de plus en plus complexes des données. Les poids des transformations linéaires sont le plus souvent trouvés de façon itérative par descente de gradients.

L'utilisation des réseaux neuronaux dans le traitement du langage naturel s'est traditionnellement faite à l'aide de *réseaux neuronaux récurrents*, souvent désignés par l'acronyme *RNN* (*Recurrent Neural Network*). Ceux-ci permettent l'évolution de cette représentation hiérarchique à travers une séquence. Une explication plus détaillée de ceux-ci sera vue au chapitre 3.

1.2 Éléments de la problématique

Dans ce document, un type de problème de recommandation spécifique sera adressé, celui de la recommandation de personne à personne. Celui-ci est différent à cause de la complexité

d'un individu, de son évolution dans le temps et de la difficulté générale de le représenter par un nombre fini de variables. Les données utilisées proviennent de E180 (2018), une plateforme permettant aux individus de se rencontrer pour apprendre l'un de l'autre.

Chaque utilisateur de la plateforme y a un certain nombre d'offres et de demandes d'expertises que les autres utilisateurs peuvent voir, dont des exemples peuvent être vus à la figure 1.1. Sur la base de celles-ci, une conversation ou une rencontre peuvent être organisées. Le but du système de recommandation est de montrer à un utilisateur les autres utilisateurs les plus intéressants sur la base de ses propres offres et demandes.

Les données présentent deux aspects : explicites et implicites. Les données explicites sont sous la forme de texte et d'étiquettes en français et/ou anglais pour chacune des offres ou demandes. Les données implicites, quant à elles, sont obtenues à partir de l'activité des individus sous la forme de vues de profil et de présence de conversations ou de rencontres.

1.3 Objectifs de recherche

L'objectif de ce mémoire est de présenter un modèle original pouvant encoder des utilisateurs dans un espace vectoriel grâce à des réseaux de neurones utilisant des données d'entrées textuelles. Cette espace permet de faire des recommandations en faisant des calculs de similarité entre utilisateurs. Le modèle combine les données collaboratives aux données textuelles à travers la fonction de perte. Pour ce faire, deux choses sont accomplies : (1) la création d'un modèle à base de réseaux neuronaux pouvant encoder chaque personne dans un même espace latent et (2), la création d'un algorithme de descente de gradients pouvant entraîner ce modèle. L'évaluation de la réussite est faite par la comparaison à un algorithme reconnu ayant eu de bons résultats sur le jeu de données (Spaeth and Desmarais (2013)) sur les métriques de précision et rappel.

La contribution principale de ce mémoire est cette nouvelle approche pour les systèmes de recommandations permettant d'entraîner des modèles utilisant des données dépendant uniquement de l'utilisateur pour faire des recommandations aussi complexes et diversifiées que celles faites par les filtres collaboratifs.

1.4 Plan du mémoire

Le restant de ce document sera présenté dans l'ordre suivant. Le chapitre 2 présentera une revue de littérature du domaine des systèmes de recommandations réciproques et de l'utilisation des réseaux neuronaux dans les systèmes de recommandations en général, le chapitre 3

expliquera le contexte mathématique nécessaire pour le chapitre 4, où le travail fait dans le contexte de la maîtrise et publié dans l'article (Lefebvre-Brossard et al., 2017) (Annexe **??**) est présenté. Finalement, la conclusion est faite au chapitre 5.

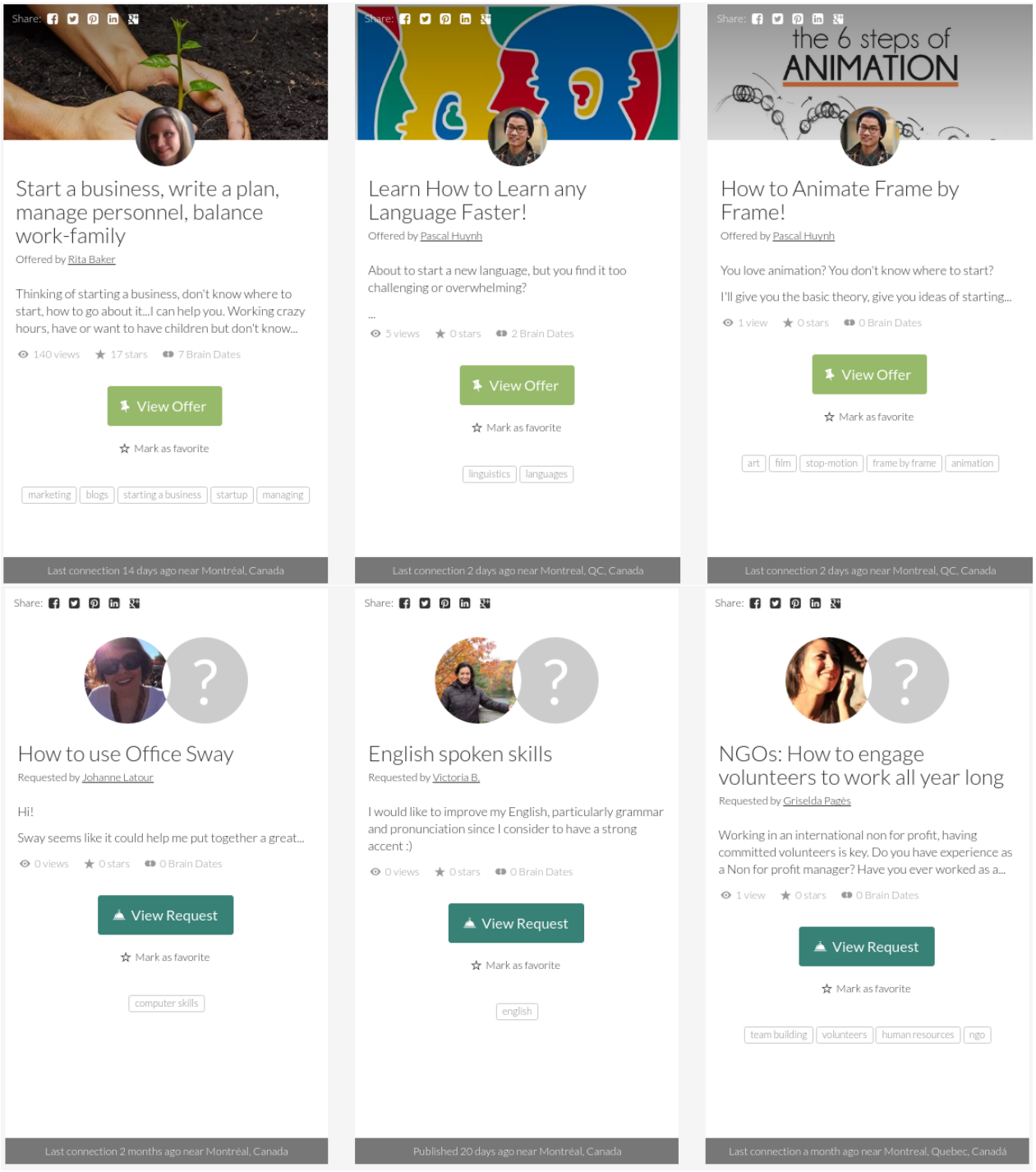


Figure 1.1 Exemples d'offres (haut) et demandes (bas)

CHAPITRE 2 REVUE DE LITTÉRATURE

Les systèmes de recommandations sont devenus de plus en plus sophistiqués avec les années, mais ce n'est que dans la dernière décennie que le besoin de systèmes de recommandations réciproques dans des systèmes avec des dizaines de milliers d'utilisateurs a mené à un essor de la rechercher dans le domaine. Parallèlement, les modèles à base de réseaux neuronaux ont fait leur preuve dans de plus en plus de domaines et récemment, ont commencé à être étudiés dans des systèmes de recommandations.

2.1 Recommandations réciproques

Pizzato et al. (2013) présentent une vue d'ensemble des systèmes de recommandations réciproques et explorent l'effet des caractéristiques de ces systèmes dans un contexte de site de rencontres. Ils décrivent les différences présentes entre les systèmes de recommandations traditionnels et les systèmes de recommandations réciproques dans la création du modèle d'utilisateur, du rôle de celui-ci, de l'évaluation du succès de la recommandation et du type d'algorithme utilisé. Ils préconisent des métriques prenant en compte différents niveaux de succès, en particulier la proportion de recommandations qui ont mené à un message de retour pour l'utilisateur à qui la recommandation a été faite dans le cas du site de rencontres. Les scores rapportés dans une étude précédente (Pizzato et al., 2010) sont discutés et montrent que prendre la moyenne harmonique du score pour l'utilisateur de la recommandation potentielle et du score où les rôles sont inversés mène à de meilleurs résultats. Ils montrent aussi que les utilisateurs ont tendance à prendre en considération les préférences d'un ou une partenaire potentielle lorsque vient le moment d'initier la communication payée, mais pas au niveau de la prise de contact gratuite, montrant qu'un utilisateur donne de l'importance à une appréciation réciproque lorsqu'il y a une pénalité potentielle. Ils observent aussi que, fidèle au stéréotype, les femmes adoptent un comportement réactif alors que les hommes adoptent un comportement proactif, mais qu'en général, le meilleur taux de succès est lorsqu'un utilisateur normalement réactif envoie un message à un utilisateur proactif alors que le pire taux est lorsqu'un utilisateur proactif envoie un message à un autre réactif. Ils évaluent aussi l'impact de l'activité et de la popularité sur le taux de succès lorsqu'une demande de contact est envoyée et montrent que les utilisateurs plus populaires ont le plus de succès, mais que pour un niveau de popularité fixé, les utilisateurs les moins actifs obtenaient le plus souvent une réponse positive, montrant peut-être ainsi qu'ils sont plus sélectifs dans les demandes qu'ils envoient. Ils discutent aussi de l'importance d'avoir un score adapté au groupe auquel

l'utilisateur appartient. Par exemple pour favoriser un utilisateur payant, on donnera une moindre importance à un nouvel utilisateur dont l'information est moins crédible ou prendra en compte la popularité des utilisateurs.

Cai et al. (2013) ont décrit un algorithme utilisant un modèle asymétrique prenant en compte des comportements positifs et négatifs. Ce modèle, ProCF, définit trois types d'entités : l'acteur, l'imitateur et le coacteur. Ce premier est l'utilisateur ayant interagi avec le coacteur et le second, celui dont le contact possible avec le coacteur doit être évalué. Pour une paire d'acteur et coacteur, deux probabilités sont définies. La positive est un ratio du nombre de coacteurs avec qui l'interaction a été positive pour les deux, sur le nombre de coacteurs avec qui l'interaction de l'acteur a été positive, mais dont l'interaction avec l'imitateur est positive ou négative. La probabilité négative est calculée de la même façon, mais en utilisant les interactions négatives. Un résidu de probabilité est ensuite calculé comme la différence de ces probabilités (positive moins négative). Lors de la recommandation, le score donné pour une recommandation est la somme des résidus de probabilités où l'imitateur est la recommandation faite et le coacteur l'utilisateur à qui la recommandation est faite. L'évaluation est faite avec des données provenant d'un site de rencontres et comparée à *Best 2CF+* (Krzywicki et al., 2010) en utilisant la précision, le rappel et le score F ainsi que le taux d'acceptation et de rejet. Les résultats montrent que ProCF performe mieux que le modèle de comparaison sur toutes les métriques.

Krzywicki et al. (2014) a décrit l'évaluation et l'implémentation d'un système de recommandations réciproques dans un contexte de site de rencontres. Puisque le besoin était d'avoir un système couvrant potentiellement quelques centaines de milliers d'utilisateurs et plusieurs millions de contacts entre ceux-ci, les méthodes étudiées devaient être extensibles à des données de cette magnitude. Ils voulaient de plus des méthodes adaptables à la nature dynamique et incrémentale de la génération de recommandations. Ils ont donc expérimenté avec quatre méthodes. La première consiste à créer une série de règles où, pour chaque attribut, la valeur optimale chez la personne du sexe opposé est trouvée par une analyse statistique (Kim et al., 2012a). La seconde méthode trouve les utilisateurs similaires à un utilisateur donné et recommande leurs contacts. Ces usagers similaires sont ceux du même sexe et orientation sexuelle dans la même catégorie d'âge (l'âge est séparé en intervalles de 5 ans) ou dans une catégorie d'âge inférieure ou supérieure, mais avec le même emplacement. Les candidats obtenus sont ensuite réordonnés en multipliant leur score par un poids fourni par des arbres de décisions entraînés sur une grande quantité de données. Cette méthode provient de Krzywicki et al. (2012). La troisième méthode combine les deux premières en utilisant les paires recommandées par la première comme contacts dans la seconde (Kim et al., 2012b). Les candidats sont réordonnés de façon similaire grâce à des arbres de décisions. La dernière

méthode est ProCF (Cai et al., 2013) décrite ci-haut, mais utilisant aussi les paires données par la première méthode comme si celles-ci étaient de vrais contacts. Pour l'évaluation de ces méthodes, chacune a reçu 10% des utilisateurs existants du site et des utilisateurs s'inscrivant pendant la période d'essai. Cette période dura 6 semaines pendant laquelle des recommandations étaient faites quotidiennement. Les métriques utilisées ont été définies en collaboration avec la compagnie et visaient à déterminer l'avantage des recommandations de chacune des méthodes par rapport à la méthode actuelle du site, c'est-à-dire laisser les utilisateurs chercher eux-mêmes des partenaires en spécifiant des critères. Les résultats ont montré que la meilleure méthode était la seconde, c'est-à-dire la recommandation de contacts provenant d'utilisateurs similaires repesés grâce à des arbres de décisions. Les chercheurs ont pu observer que les résultats obtenus sur des données historiques ne reflètent pas nécessairement le comportement sur un système réel. Ceci provient du fait que l'évaluation d'un modèle sur des données historiques permet de trouver le meilleur modèle pouvant prévoir le comportement d'un utilisateur n'utilisant pas de système de recommandations, mais que ce résultat ne sera pas nécessairement le même pour un système déployé où le but est de changer le comportement de l'utilisateur. Cependant, les chercheurs ont montré dans Krzywicki et al. (2012) que l'évaluation sur des données historiques pouvant tout de même prévoir les tendances d'une vraie situation.

Zhao et al. (2014) étudient aussi la recommandation d'utilisateurs dans un contexte de site de rencontres. À partir de leurs données, ils créent un tenseur dont les dimensions représentent les utilisateurs masculins, les utilisateurs féminins et une troisième dimension de taille 2, où le premier élément représente un contact (initial ou réciproque) de la part de l'utilisateur masculin et le second la même chose, mais venant de l'utilisateur féminin. Partant de ce tenseur, ils définissent la similarité entre deux utilisateurs comme une combinaison du goût (attirance pour des utilisateurs similaires) et de l'attrait (attirance par des utilisateurs similaires) de ceux-ci. Le score de la recommandation est une combinaison de cette similarité et d'une pénalité déterminée par l'absence ou la présence d'un échange réciproque entre cet individu similaire et l'usager recommandé. Les recommandations ont été évaluées par la précision et le rappel sur les contacts initiés et sur les contacts réciproques. Ces résultats ont été comparés à des filtres collaboratifs basés sur 2 différentes matrices de contacts. La première contenait des éléments valant 1 si un contact était initié par l'un ou l'autre des utilisateurs alors que la seconde n'avait cet élément valant 1 que lorsque le contact était réciproque. Le modèle a aussi été comparé à CCR (Akehurst et al., 2011), qui utilise sept attributs des utilisateurs pour déterminer leur similarité et utilise leurs interactions passées pour faire les recommandations, et pLSA (Hofmann, 2004), une méthode créant des variables latentes permettant de classifier les individus dans différentes catégories et de faire les recommandations basées sur

celles-ci. Les résultats montrent que leur modèle performe mieux sur toutes les métriques sauf pour la précision chez les utilisateurs féminins lorsque peu de recommandations sont faites, montrant que considérer à la fois le goût et l'attrait de chacun aide à faire de meilleures recommandations.

Dans Prabhakar et al. (2017), les chercheurs ont étudié la question d'utiliser un système de recommandations pour la recommandation de pairs pour un cours en ligne ouvert et massif (*Massive Open Online Course*). Comme données, ceux-ci ont utilisé celles issues des cours en lignes sur la plateforme edX provenant de MITx et HarvardX. Ils ont filtré celles-ci pour ne garder que les observations où l'âge, la location, le diplôme et le sexe étaient présents. De plus, ils ont ajouté des données artificielles sur les intérêts des apprenants. Lorsqu'un individu s'inscrit sur la plateforme, il définit les préférences qu'il a pour ces attributs. Une matrice de distance est ensuite bâtie en prenant, pour chaque individu, la moyenne des distances entre ses préférences et les attributs de tous les autres usagers. L'âge et le diplôme sont chacun divisés en 5 catégories et la distance calculée en prenant leur différence absolue et en normalisant pour que celle-ci soit entre 0 et 1. Pour le sexe et la location, la distance est 0 s'ils sont identiques et 1 si différents. Dans le cas des intérêts, cet attribut est défini comme une variable hiérarchique, laquelle hiérarchie est définie en utilisant WordNet (Miller, 1995). La similarité est calculée en adaptant la méthode de Wu and Palmer (1994). Pour chaque paire d'individus, le score réciproque est calculé comme la moyenne harmonique entre leur score l'un pour l'autre dans cette matrice. Une liste ordonnée en fonction de ce score est créée pour chaque individu et les recommandations faites en prenant les k individus avec les plus petites distances. Une recommandation est considérée bonne si les deux individus concernés sont dans leur liste respective. Cette définition est utilisée pour calculer la précision et le rappel. De plus, les auteurs utilisent une version modifiée de *Discounted Cumulative Gain* (Järvelin and Kekäläinen, 2002) pour évaluer l'ordonnancement fait par l'algorithme. Les chercheurs comparent à une méthode n'utilisant pas l'aspect réciproque du score et montrent que leur algorithme performe mieux que le modèle de base sur toutes les métriques.

Potts et al. (2018) ont aussi créé un système de recommandations pour la recommandation de pairs dans le but de favoriser l'apprentissage. Pour ce faire, ils ont bâti une plateforme où les étudiants peuvent mettre les plages horaires durant lesquelles ils sont disponibles. Leurs compétences sont évaluées par des questionnaires à choix multiples offerts sur la plateforme. Ces compétences peuvent aussi être modifiées en fonction de la performance de l'étudiant sur des travaux et des tests faits sur celle-ci. Le but du système est de recommander des utilisateurs pour une liste de sujets et de rôles (tuteur, apprenant ou partenaire d'étude) à une période donnée. Un premier élément du score pour une possible session est donné par la fonction logistique de la norme d'un vecteur, dont les deux éléments sont la compétence de

chacun sur le sujet de la session, mis à la puissance d'un hyperparamètre τ, définissant le seuil au-delà duquel la session sera probablement fructueuse, et multiplié par un second hyperparamètre $\frac{1}{\alpha}$ permettant de modifier l'indulgence de la combinaison de ces deux utilisateurs. Le second élément est une mesure de la préférence de l'utilisateur à qui la recommandation est faite pour la différence de compétences. Cet élément est défini comme la densité d'une gaussienne au point de la compétence de l'utilisateur recommandé avec comme moyenne la somme de la compétence de l'utilisateur à qui la recommandation est faite et d'une mesure de sa préférence. Le score final est le produit de ces deux facteurs, agrégés sur les sujets et rôles voulus par l'utilisateur à qui la recommandation est faite. Les recommandations sont faites basées sur un score réciproque, calculé comme la moyenne harmonique entre les scores des deux utilisateurs l'un pour l'autre. Les métriques utilisées pour l'évaluation sont l'extensibilité (le temps pris pour rouler l'algorithme), la précision réciproque (les recommandations sont dans les deux listes respectives), la couverture (la proportion d'utilisateurs recommandés à au moins un autre) et la qualité (la moyenne des compétences des utilisateurs appariés sur tous les sujets). Les données utilisées ont été créées artificiellement. La précision a été comparée avec des recommandations faites seulement avec le score de l'utilisateur à qui elles sont faites et évaluée avec différents nombres de recommandations. La couverture et la qualité ont été évaluées en modifiant certains hyperparamètres. Le nombre total d'utilisateurs a un effet important sur l'extensibilité du modèle, ainsi que le nombre de recommandations faites. L'utilisation du score réciproque a un effet important sur la précision réciproque dont la différence augmente avec le nombre de recommandations et reste relativement constante avec le nombre d'utilisateurs. Augmenter le nombre de recommandations ainsi que diminuer le seuil τ permet d'augmenter la couverture. Finalement, un haut seuil τ ainsi qu'une basse indulgence $\frac{1}{\alpha}$ augmentent la qualité des recommandations.

2.2 Systèmes de recommandations basés sur les réseaux de neurones

Les réseaux de neurones ont apporté des avancées importantes à plusieurs domaines d'application et il n'est pas surprenant que la communauté des systèmes de recommandations s'y soit aussi penchée.

Liang et al. (2016) s'inspire des succès obtenus par les modèles utilisant des représentations vectorielles de mots (word2vec, GloVe, etc.) pour améliorer un système de recommandations utilisant une factorisation de matrices avec des filtres collaboratifs. Cette factorisation décompose la matrice de scores en un produit de facteurs latents pour les utilisateurs et les items.

$$\hat{R}_{ui} = \boldsymbol{\theta}_u^T \boldsymbol{\beta}_i$$

La fonction objective est une combinaison de la différence entre l'approximation et la valeur réelle, et une régularisation L_2 des facteurs latents pour les utilisateurs et les items.

$$L_{MF} = \sum_{u,i} c_{ui} \left(R_{ui} - \boldsymbol{\theta}_u^T \boldsymbol{\beta}_i \right)^2 + \lambda_\theta \sum_u \|\boldsymbol{\theta}_u\|_2^2 + \lambda_\beta \sum_i \|\boldsymbol{\beta}_i\|_2^2$$

De plus, la matrice SPPMI (*shifted positive pointwise mutual information*) est calculée, représentant les patrons de co-occurrence entre chaque item et son contexte. Celle-ci est obtenue par

$$\text{SPPMI}(i,j) = \max\left(\text{PMI}(i,j) - \log k, 0\right)$$
$$\text{PMI} = \log \frac{\#(i,j) \cdot D}{\#(i) \cdot \#(j)}$$

où $\#(i,j)$ représente le nombre de fois où j apparaît dans le contexte de i, $\#(i) = \sum_j \#(i,j)$, $\#(j) = \sum_i \#(i,j)$ et D est le nombre de paires de mot et contexte. Ce contexte est défini ici comme tous les items apparaissant dans l'historique d'un usager. À la fonction de coût précédente est ajouté un élément donnant la différence entre cette matrice de co-occurrence et une combinaison linéaire des représentations d'item par la factorisation de matrice et les représentations vectorielles du contexte. L'entraînement se fait par une méthode similaire aux moindres carrés alternés (*Alternating Least Squares* ou ALS) jusqu'à convergence. La méthode est évaluée sur 3 jeux de données avec les métriques de rappel, de précision et NDCG (*truncated normalized discounted cumulative gain*), une mesure de la justesse de l'ordonnancement. Le modèle de base auquel l'algorithme est comparé est la factorisation de matrice pesée (Hu et al., 2008). Le modèle des auteurs, CoFactor, performe systématiquement mieux que le modèle de base sur tous les jeux de données et toutes les métriques.

Wang et al. (2015) propose une des premières méthodes alliant les systèmes de recommandations avec les réseaux neuronaux. Ils présentent un modèle hiérarchique bayésien composé de deux parties. La première crée une représentation vectorielle de l'item grâce à un *Stacked Denoising Autoencoder* (SDAE), un réseau de neurones encodant une version corrompue des données d'entrées dans une représentation plus petite puis utilisant celle-ci comme entrée d'un décodeur reconstruisant les données de départ. La représentation latente donnée par le décodeur devient la représentation de l'item. La représentation de l'utilisateur est échantillonnée d'une multinormale. Le score de l'utilisateur pour l'item est calculé comme le produit vectoriel de ces deux représentations latentes. Les paramètres sont entraînés par descente de gradients, les représentations latentes des items et utilisateurs par descente de coordonnées et les différents hyperparamètres en utilisant un ensemble de validation et en faisant une recherche par quadrillage sur ceux-ci. Le modèle est évalué sur deux jeux de données

de citations d'articles et un troisième provenant d'un concours présenté par Netflix [1]. Les données d'entrée pour les deux premiers jeux sont les titres et résumés alors que pour celui de Netflix, ce sont les résumés de films pris de IMDB [2]. Dans tous les cas, ces données sont transformées en sac de mots avant d'être traitées par le SDAE. La comparaison est faite avec d'autres systèmes de recommandations hybrides et montre que le modèle des auteurs performe systématiquement mieux que les autres.

Dans Zhang et al. (2017), les chercheurs font une revue de l'utilisation des réseaux de neurones pour les systèmes de recommandations. Ils ont séparé les différents modèles en deux facteurs, soit le type de réseau de neurones utilisé et l'intégration avec les modèles plus classiques de systèmes de recommandations. Ils créent une subdivision du type de réseau de neurones en séparant en deux catégories les différents algorithmes sur la base de l'utilisation d'un seul type de réseau de neurones ou de la combinaison de plusieurs. L'intégration est déterminée par le fait que le réseau de neurones est utilisé avec des approches traditionnelles ou par lui-même. Les articles étudiés utilisent des systèmes de recommandations dans plusieurs domaines tels que la recommandation de musique, de nouvelles ou de citation, mais sont tous dans la catégorie générale de recommandation d'item (objet sans opinion ou préférence) à des personnes (objet avec des préférences). (He et al., 2017; Wang et al., 2017; Lian et al., 2017) utilisent des perceptrons multicouches pour encoder les utilisateurs et items séparément avant de les combiner avec un autre perceptron multicouches. (Cheng et al., 2016; Chen et al., 2017a) étendent ce modèle pour ajouter un perceptron multicouches connectant directement la dernière couche avec certains attributs n'ayant pas besoin d'être généralisés, mais plutôt mémorisés. Cette approche nécessitant de la sélection de variables, Guo et al. (2017) propose une approche combinant la machine de factorisation et les perceptrons multicouches, où les perceptrons modélisent les interactions de haut niveau entre les usagers et les items et la machine de factorisation modélise celles de bas niveau. Chen et al. (2017b) améliore les filtres collaboratifs traditionnels en appliquant un mécanisme d'attention, composé d'un perceptron multicouches, pour sélectionner les items les plus représentatifs pour chaque utilisateur et les attributs d'item les plus importants pour chaque usager. Alashkar et al. (2017) utilisent deux perceptrons multicouches pour modéliser les données et les règles d'experts, et entraînés en minimisant la distance entre leurs sorties, permettant ainsi de profiter des connaissances d'experts. D'autres approches utilisent des autoencodeurs, ceux-ci pouvant être utilisés de deux façons : en utilisant une couche intermédiaire plus petite comme représentation latente ou en reconstruisant une matrice de scores. Sedhain et al. (2015) utilise la seconde approche en entraînant un modèle pour reproduire les vecteurs d'utilisateurs ou d'items dans la matrice

1. https ://www.netflix.com
2. https ://www.imdb.com/

de scores. Strub et al. (2016) étendent cette méthode en utilisant une version où les données d'entrées sont corrompues et des informations sur les attributs d'utilisateur et d'items sont incorporées, rendant le modèle plus robuste et mitigeant le problème de manque de données de départ (*cold start problem*). Ying et al. (2016) utilise un modèle alliant une approche bayésienne avec un autoencodeur pour créer une liste ordonnée de recommandations basée sur les différences de préférence entre items dans une paire. Gong and Zhang (2016) utilise des réseaux de neurones convolutionnels (CNN) avec attention pour recommander des hashtags en utilisant le texte d'un tweet comme données. L'attention est utilisée pour seulement donner de l'importance aux mots les plus informatifs. Zheng et al. (2017) utilise deux CNNs pour encoder le comportement des utilisateurs et les attributs des items en se basant sur le texte de critiques avant de les combiner par une machine de factorisation à la dernière couche. Hidasi et al. (2015) utilise un réseau de neurones récurrents (RNN) pour recommander la prochaine action d'un utilisateur en utilisant ses actions précédentes. Hidasi et al. (2016) étendent ce modèle en utilisant de l'information additionnelle pour créer des RNNs encodant chacun un type de données avant de concaténer les résultats pour prédire le prochain item. Smirnova and Vasile (2017) utilisent aussi de l'information additionnelle, mais l'utilisent directement comme entrée d'un RNN prédisant les clics suivants. Wu et al. (2017) utilise un RNN pour le score donné par un utilisateur à un item, en prenant compte que l'attrait d'un item et les préférences d'un utilisateur évoluent au fil du temps. Li et al. (2016) allie un mécanisme d'attention avec un RNN pour prédire les hashtags à partir d'un texte. LDA est utilisé pour trouver les sujets des différentes publications et l'attention est calculée en utilisant ces distributions de sujets.

Beutel et al. (2018) introduisent une nouvelle méthode d'incorporation d'informations additionnelles provenant du contexte à un système de recommandations utilisant des réseaux de neurones. Pour ce faire, toute source d'information additionnelle est transformée en un vecteur, puis elles sont additionnées ensemble avec une constante de 1 ajoutée. Ce vecteur est ensuite multiplié par élément avec soit l'entrée ou la sortie d'un RNN. Ceci est interprétable comme étant un masque ou un mécanisme d'attention sur ces sources d'information additionnelle et permet de modéliser plus facilement les relations entre les entrées et ces sources. Ils testent le modèle en utilisant un jeu de données contenant les séquences de visionnement de vidéos pour des centaines de millions d'utilisateurs et comparent aux modèles de base en utilisant la métrique de précision moyenne pour une recommandation ou vingt. Les résultats montrent que l'utilisation de cette méthode, que les auteurs appellent *Latent Cross*, est une amélioration comparée aux modèles de base.

Liu et al. (2018) utilisent l'information explicite des utilisateurs et items pour prédire si un utilisateur a évalué un item. Leur modèle est séparé en deux étapes. Dans un premier temps,

un autoencodeur tente de reconstruire les attributs d'un utilisateur et la couche intermédiaire est concaténée à un *embedding* de l'utilisateur, c'est-à-dire un vecteur le représentant, dont les éléments sont obtenus lors de l'entraînement. La même chose est faite pour l'item avec un autoencodeur et un *embedding* différents. Par la suite, deux méthodes différentes sont utilisées pour prédire la sortie. La première, GMF++, multiplie ces deux vecteurs par éléments avec de passer le résultat dans un perceptron multicouches. La seconde, MLP++, les concatène plutôt que les multiplier. La fonction de perte est une somme de la fonction de perte de chacun des deux autoencodeurs et la perte par entropie croisée entre la sortie du modèle et l'existence réelle ou pas d'une évaluation de l'item par l'utilisateur. Le jeu de données sur lequel ce modèle est évalué est comporte les évaluations d'utilisateurs pour des films. L'évaluation est faite en tentant de prédire la dernière évaluation d'un utilisateur en utilisant le restant de ces évaluations. Les métriques utilisées sont le *Hit Ratio* (si l'item testé est dans les k premières recommandations) et le *Normalized Discounted Cumulative Gain* (une mesure prenant en compte la position de l'item dans les k recommandations). Puisque des expériences dans un article précédent montrent que le modèle original GMF performe mieux que le modèle original MLP, seul GMF++ a été comparé aux modèles de référence. Les résultats montrent que le nouveau modèle performe systématiquement mieux que ceux de référence.

CHAPITRE 3 CONTEXTE

Pour bien comprendre le modèle présenté au chapitre suivant, nous allons réviser certaines notions utiles. Ce chapitre sera séparé en deux parties, soit les systèmes de recommandation et les réseaux neuronaux.

3.1 Systèmes de recommandations

Le but d'un système de recommandation est de donner un choix plus restreint d'éléments pertinents pour un utilisateur. Celui-ci n'a probablement ni le temps, ni l'envie de passer par-dessus tous les items disponibles et un nombre trop important de choix peut le conduire à ne pas prendre une décision judicieuse (Tierney (2011)) ou à subir du mécontentement à cause de cette quantité (Iyengar and Lepper (2000)). On peut penser par exemple à quelqu'un qui veut écouter un film pour relaxer ne sachant pas lequel choisir ou le mauvais objet acheté en ligne parce que la recherche pour le bon aura été trop longue.

La qualité des systèmes de recommandations est en général en général évaluée selon quatre critères :

Pertinence : le facteur de succès le plus important d'un système de recommandations est la pertinence, c'est-à-dire le filtrage de tous les items possibles pour recommander ceux qu'un usager trouvera les plus intéressants. Toute la difficulté du problème provient de cette découverte de ce qu'un utilisateur considèrera intéressant.

Nouveauté : un utilisateur à qui on ne recommandera que des items qu'il a déjà achetés ou consommés se désintéressera bien vite du système de recommandations. Pour éviter cela, le système devra être en mesure de recommander de nouveaux items. Toutefois, pour certains contextes, recommander des items déjà achetés peut être positif ; on peut penser à des lames de rasoir qui doivent être remplacées fréquemment.

Sérendipité : la sérendipité d'une recommandation s'apparente à l'expérience d'un heureux hasard, c'est-à-dire que l'utilisateur n'aurait jamais pensé à chercher cet item, mais qu'il trouve celui-ci intéressant. La sérendipité diffère de la nouveauté par le fait que, contrairement à cette dernière, elle procure un effet de surprise. Par exemple, si une utilisatrice aimant beaucoup les films d'amour se fait recommander le nouveau film sortant à la Sainte Valentin, il y aura présence de nouveauté, mais pas de sérendipité. Mais si elle voit sur sa liste, *Mr. Nobody*, un film de science-fiction tournant autour d'une relation amoureuse, celui-ci sera perçu comme un heureux hasard si elle

n'a pas l'habitude d'écouter de la science-fiction.

Diversité : la diversité est le fait que toutes les recommandations faites ne sont pas similaires, mais varient légèrement sur différents aspects que l'utilisateur trouve intéressants. Par exemple, un utilisateur utilisant un système de recommandations pour trouver des activités pourrait apprécier de se faire suggérer à la fois des groupes de lectures et des excursions en montagne.

La façon dont un système de recommandations offre ses suggestions se décline de deux façons : évaluer le score qu'un utilisateur mettrait à chaque item ou ordonner la liste des items dans l'ordre de leur pertinence (Aggarwal et al., 2016, chap 1.2).

La première approche peut être vue comme un problème d'imputation de valeurs manquantes dans une matrice de scores d'utilisateurs. Pour u utilisateurs et n items, cette matrice $\boldsymbol{R}$ sera de taille $u \times i$ avec chaque élément R_{ij} correspondant au vote de l'utilisateur i pour l'item j. Le but du problème devient donc d'utiliser les valeurs déjà présentes et possiblement d'autres données externes pour prédire les valeurs manquantes de cette matrice.

La deuxième consiste à créer une liste de recommandations avec un ordre total ou partiel, c'est-à-dire ordonner tous les items possibles ou simplement retourner les k meilleurs dans un ordre arbitraire. Bien que, souvent, cette liste est créée à partir de la première approche, elle peut aussi être faite à partir de mesures de similarité qui ne dépendent pas d'un score quelconque. On peut penser par exemple à la recommandation d'un film basée sur la similarité de sa description avec celle de films précédemment vus par l'utilisateur.

3.1.1 Filtres collaboratifs

Les systèmes de recommandations basés sur les filtres collaboratifs utilisent l'historique des interactions des utilisateurs pour faire des recommandations. Ils se basent sur le fait que deux utilisateurs ayant eu un comportement similaire par le passé ont de bonnes chances d'avoir un comportement similaire dans le futur ou que deux items ayant été évalués similairement par des individus risquent d'avoir des scores similaires dans le futur. Ces deux approches sont la base des deux principales familles de filtres collaboratifs.

Filtres collaboratifs par utilisateur (*user-user*) : Pour évaluer un nouvel item pour un utilisateur donné, on calculera d'abord les similarités entre cet utilisateur et ceux ayant déjà évalué l'item. Ces similarités serviront ensuite de poids pour calculer une moyenne pondérée des évaluations de l'item, ce qui sera le score prédit de cet item par l'utilisateur.

Filtres collaboratifs par item (*item-item*) : L'approche par item fait un calcul si-

milaire à l'approche *user-user*, mais en utilisant une similarité entre item plutôt qu'entre utilisateurs. Le score prédit sera la moyenne des votes de l'utilisateur à ces items pondérés par leur similarité.

Indicateurs de scores

Bien que le mot score ou évaluation est utilisé, le score donné par un utilisateur à un item n'est pas nécessairement explicite. Un utilisateur peut donner une évaluation implicite à un item simplement en interagissant avec, par exemple en cliquant dessus, en l'achetant ou en le visionnant. On peut penser par exemple aux recommandations faites par Youtube (Davidson et al. (2010)) basées en partie par les visionnements faits par l'utilisateur. Le mot score utilisé pour le restant du document est un nombre quelconque représentant une combinaison de différents facteurs implicites et/ou explicites.

Comme vu précédemment, il est possible de représenter les évaluations faites par les utilisateurs pour des items dans une matrice $\boldsymbol{R} \in \mathbb{R}^{u \times n}$ où u est le nombre total d'utilisateurs et n le nombre total d'items et chaque élément R_{ij} représente le score donné à l'item j par l'utilisateur i. Puisque seulement une minorité d'utilisateurs auront évalué une minorité d'items, cette matrice sera typiquement très creuse.

Ces scores peuvent être de plusieurs types :

Variables indicatrices (entiers positifs) : Le score indique simplement le nombre d'interactions avec l'item, le nombre de visionnements d'un vidéo par exemple, ou l'appréciation positive de celui-ci sans possibilité d'une évaluation négative. L'action d'aimer (*like*) sur Facebook en est un exemple. Chaque élément de la matrice sera donc un 1 si l'item a été apprécié. On utiliser 0 si l'item n'a pas été vu ou qu'il est ignoré.

Nombres binaires : L'item peut soit avoir un score positif ou négatif. Par exemple, les évaluations faites sur Netflix permettent uniquement de donner un pouce vers le haut ou vers le bas aux films écoutés. Ces scores seront typiquement représentés dans la matrice par un -1 pour les scores négatifs, 1 par les scores positifs et 0 pour les items n'ayant pas encore été évalués

Nombres entiers (ou facteurs) dans un intervalle : Plutôt que d'être limité par un score positif ou négatif comme avec le nombre binaire, un nombre dans un intervalle permet une évaluation plus globale. L'utilisation de cinq étoiles pour évaluer un produit en est un exemple tout comme l'utilisation d'un nombre sur dix. Il est à noter que cette catégorie englobe aussi l'utilisation de demi-mesure comme une demi-étoile puisque celle-ci peut être transformée en un intervalle de nombres entiers. On voit

aussi souvent l'utilisation d'intervalles composés d'expressions comme «je suis en accord» ou «je suis très en désaccord». Lorsque ces expressions forment un ordre, il est possible de les transformer en nombres et de les utiliser de la même façon. Ces scores sont en général représentés dans la matrice en utilisant le 0 comme non-évaluation et en commençant l'intervalle à 1.

Nombres réels : Bien que moins utilisés, il est possible de représenter les scores par des nombres réels. La raison pour leur relative rareté est l'augmentation du nombre de choix possibles pour l'utilisateur sans que cela leur permette de communiquer une information plus complexe. L'ajout de précision décimale à un score n'aura en général que peu d'impact par rapport à la partie entière (ou à la moitié si la possibilité existe) et ne fait qu'apporter plus de complexité au modèle. Si l'intervalle comprend la valeur 0, le problème de représentation de valeurs manquantes vient aussi ajouter de la complexité inutile.

Pour la suite de ce document, nous assumerons que les scores sont des nombres entiers positifs et que les valeurs manquantes sont remplacées par 0.

Calculs de similarité

Il existe plusieurs métriques différentes pour calculer la similarité entre deux utiliseurs, mais nous verrons ci-dessous deux des plus populaires : le coefficient de corrélation de Pearson et la similarité du cosinus.

Le *coefficient de corrélation de Pearson* est une mesure de la corrélation linéaire entre deux variables, c'est-à-dire à quel point la variation d'une des variables est accompagnée de la variation de l'autre. Pour deux utilisateurs u et v (lignes) dans la matrice $\boldsymbol{R}$, le coefficient est défini par :

$$\begin{aligned} w_{uv} &= \frac{\sum_i (R_{ui} - \mu_u)(R_{vi} - \mu_v) \mathbb{1}_{\{R_{ui} \neq 0, R_{vi} \neq 0\}}}{\sqrt{\sum_i (R_{ui} - \mu_u)^2 \mathbb{1}_{\{R_{ui} \neq 0, R_{vi} \neq 0\}}} \sqrt{\sum_i (R_{ui} - \mu_u)^2 \mathbb{1}_{\{R_{ui} \neq 0, R_{vi} \neq 0\}}}} \\ \mu_u &= \frac{\sum_i R_{ui} \mathbb{1}_{\{R_{ui} \neq 0, R_{vi} \neq 0\}}}{\sum_i \mathbb{1}_{\{R_{ui} \neq 0, R_{vi} \neq 0\}}} \\ \mu_v &= \frac{\sum_i R_{vi} \mathbb{1}_{\{R_{vi} \neq 0, R_{vi} \neq 0\}}}{\sum_i \mathbb{1}_{\{R_{vi} \neq 0, R_{vi} \neq 0\}}} \end{aligned} \tag{3.1}$$

On remarquera que les variables indicatrices font en sorte que seuls les items évalués par les deux utilisateurs sont utilisés dans le calcul. Ceci cause un problème s'il n'existe que peu d'items en communs entre deux individus parmi ceux qu'ils ont évalués. Si ce faible nombre de scores est identique ou très semblable, la corrélation sera plus grande que pour deux

utilisateurs ayant beaucoup de scores en communs, mais qui sont un peu moins semblables. Prenons 3 individus ayant évalués les films suivants :

Tableau 3.1 Exemple d'évaluation de films

	Blade Runner	Arrival	Star Wars	The Notebook	Romeo + Juliet	Spider Man	Mr. Nobody	Gone With the Wind
Utilisateur 1	4	4	3	0	0	2	0	5
Utilisateur 2	0	0	3	5	5	2	2	5
Utilisateur 3	5	5	4	3	3	0	5	5

Bien que l'utilisateur 3 soit clairement plus semblable à l'utilisateur 1, leur coefficient de corrélation sera de 0.816 alors que celui entre les utilisateurs 1 et 2 sera de 1 (*n.b.* les 0 sont considérées comme des valeurs manquantes pour ce calcul). Pour régler ce problème, d'autres mesures ont été créées, dont la similarité du cosinus.

Cette *similarité du cosinus* détermine la similarité entre deux utilisateurs en calculant le cosinus de l'angle entre les vecteurs de leurs éléments. Pour deux utilisateurs u et v (lignes) dans la matrice $\boldsymbol{R}$, celle-ci est calculé par :

$$\begin{aligned} w_{uv} &= \frac{\sum_i R_{ui} R_{vi}}{\sqrt{\sum_i R_{ui}^2}\sqrt{\sum_i R_{vi}^2}} \\ &= \frac{\boldsymbol{R}_u^T \boldsymbol{R}_v}{\|\boldsymbol{R}_u\|\|\boldsymbol{R}_v\|} \end{aligned} \tag{3.2}$$

Cette formule permet de corriger le problème du poids trop important donné aux concordances représentants une petit partie des scores donnés en normalisant les vecteurs des individus. En faisant ceci, un grand nombre d'éléments presque identiques aura un plus grand poids qu'un petit nombre d'éléments identiques sur la similarité finale.

Grâce à cette formule, la similarité entre les utilisateurs 1 et 2 n'est que de 0.474 alors qu'elle est de 0.795 entre les utilisateurs 1 et 3.

Cette métrique a permis de résoudre le problème de l'importance démesurée donnée lorsqu'il y a peu d'éléments en commun, mais elle a perdu l'avantage que la corrélation avait de

centrer les éléments. La *similarité du cosinus ajustée* vient régler cela en combinant les deux approches :

$$
\begin{aligned}
w_{uv} &= \frac{\sum_i \left(R_{ui} - \mu_u \mathbb{1}_{\{R_{ui}\neq 0\}}\right)\left(R_{vi} - \mu_v \mathbb{1}_{\{R_{vi}\neq 0\}}\right)}{\sqrt{\sum_i \left(R_{ui} - \mu_u \mathbb{1}_{\{R_{ui}\neq 0\}}\right)^2}\sqrt{\sum_i \left(R_{vi} - \mu_v \mathbb{1}_{\{R_{vi}\neq 0\}}\right)^2}} \\
\mu_u &= \frac{\sum_i R_{ui}}{\sum_i \mathbb{1}_{\{R_{ui}\}}} \\
\mu_v &= \frac{\sum_i R_{vi}}{\sum_i \mathbb{1}_{\{R_{vi}\}}}
\end{aligned}
\tag{3.3}
$$

On peut voir que cette formule est identique à la corrélation sauf pour le fait que plutôt que considérer seulement les paires pour le calcul des moyennes et écarts-types, on considère tous les items que l'un des deux utilisateurs aura évalués.

L'approche item-item est identique à l'approche utilisateur-utilisateur, mais en comparant les colonnes de la matrice $\boldsymbol{R}$ plutôt que les lignes.

L'approche par filtres collaboratifs présente quelques problèmes. Le plus important est qu'il nécessite un historique important avant de pouvoir donner des résultats plausibles. Ceci est est appelé le *cold start problem*. Celui-ci concerne autant un nouvel utilisateur qu'un nouvel item. Dans le premier cas, le problème est moins important puisqu'une certaine diversité des items les plus populaires peuvent simplement être recommandés et à partir de l'évaluation de ceux-ci, des recommandations plus précises peuvent être faites. Cependant, dans le second cas, les items proposés ont moins de chances d'être proposés et d'ainsi bâtir un historique de scores. Certaines solutions ont été proposées, comme par exemple l'apprentissage actif (Elahi et al. (2016)) et plusieurs solutions utilisant des approches hybrides avec les systèmes à base de contenu (Burke (2002)).

3.1.2 Systèmes à base de contenu

Contrairement aux filtres collaboratifs, les systèmes à base de contenu n'utilisent pas l'information provenant de l'historique de scores de tous les utilisateurs, mais seulement l'information concernant l'utilisateur visé et celle des items à recommander. Par exemple, un film pourrait être recommandé parce qu'il a le même réalisateur que plusieurs d'autres films appréciés par l'utilisateur.

Lorsqu'il y a de l'information disponibles sur les items, l'historique de l'utilisateur devient suffisant pour faire des recommandations puisque ses préférences pour des items potentiels

peuvent être déterminées par la similarité de ceux-ci avec les items qu'il a apprécié dans le passé.

Cette approche règle le problème de *cold start* pour les nouveaux items puisque leur évaluation pour un utilisateur ne dépend pas d'évaluations antérieures par d'autres utilisateurs, mais seulement de leur ressemblance avec les items déjà évalués. Par exemple, pour un système de recommandations d'articles à évaluer, un nouvel article sera recommandé si le sujet est similaire à d'autres articles évalués par l'utilisateur.

Un système de recommandations basé sur le contenu pourra utiliser un, ou les deux types d'information : l'information sur les items et l'information sur les préférences des utilisateurs.

Cette première information comprend tout attribut jugé utile sur un item. Ceci peut être un attribut dépendant uniquement de l'item, comme son prix ou sa description dans le cas d'un produit sur une boutique en ligne, ou l'âge et la distance dans le cas d'un site de rencontre. Cependant, celui-ci peut aussi être un attribut défini par les autres usages comme le score moyen ou les critiques données par ceux-ci.

L'information sur les utilisateurs, quant à elle, peut être explicite ou implicite, c'est-à-dire qu'elle peut être fournie par l'utilisateur lui-même ou obtenu en observant ses interactions avec le système. Par exemple, dans un contexte de site de rencontre, l'information explicite serait la demande de conversation qu'un utilisateur fait à un autreet les informations personnelles qu'il entre sur le site, et l'information implicite serait les profils que l'utilisateur visite.

Contrairement aux filtres collaboratifs, l'information utilisée pour les systèmes à base de contenu est souvent non-structurée et doit être transformée avant de pouvoir être utiliser dans le système de recommandations. Cette information est souvent sous forme de texte et doit être convertie sous forme numérique avant d'être utilisable. Un exemple de ceci est la conversion de ce texte par TF-IDF, présenté à la section 4.2.1, une technique pour transormer un bloc de texte en un vecteur où chaque élément représente la fréquence relative du mot correspondant dans le texte.

3.1.3 Réduction de dimensions

Les approches vues précédemment souffrent toutes deux du problème du rareté des données. Même si deux items ou deux utilisateurs sont similaires, ils risquent de ne pas posséder exactement les mêmes attributs, mais des attributs semblables. Une façon de réduire ce problème est de plutôt utiliser les facteurs latents de ceux-ci. Plusieurs méthodes pour obtenir ces facteurs latents existent, mais nous décrirons ici la méthode SVD (*Singular-Value Decomposition*).

Toute matrice $\boldsymbol{R} \in \mathbb{R}^{u \times i}$ peut être décomposée de la façon suivante :

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

avec $\boldsymbol{U} \in \mathbb{R}^{u \times u}$ les vecteurs propres et orthonormaux de $\boldsymbol{R}\boldsymbol{R}^T$, $\boldsymbol{V} \in \mathbb{R}^{i \times i}$ les vecteurs propres et orthonormaux de $\boldsymbol{R}^T\boldsymbol{R}$, et $\boldsymbol{\Sigma} \in \mathbb{R}^{u \times i}$ la matrice diagonale contenant les racines carrées des valeurs propres de $\boldsymbol{R}$ (see Lay et al., chap 7.4).

Ces matrices peuvent être utilisées de deux façons principales dans le contexte des systèmes de recommandations : en approximant la matrice $\boldsymbol{R}$ ou en créant une nouvelle base vectorielle de facteurs latents.

La matrice $\boldsymbol{R}$ peut être approximée par

$$\boldsymbol{R}' = \boldsymbol{U}_d\boldsymbol{\Sigma}_d\boldsymbol{V}_d^T$$

où $\boldsymbol{U}_d \in \mathbb{R}^{u \times d}$, $\boldsymbol{V}_d \in \mathbb{R}^{i \times d}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ et $d < \min(u, i)$. $\boldsymbol{U}_d$, $\boldsymbol{V}_d$ et $\boldsymbol{\Sigma}_d$ représentent respectivement les d premiers vecteurs propres des colonnes, vecteurs propres des lignes et valeurs propres. $\boldsymbol{R}'$ est une matrice dense où chaque élément peut être pris comme une prédiction de l'item de la colonne correspondante pour l'utilisateur de la ligne correspondante.

L'autre utilisation de la décomposition SVD est de créer une nouvelle base vectorielle de facteurs latents des colonnes (ou lignes) obtenue par $\boldsymbol{U}_d\boldsymbol{\Sigma}_d$ (ou $\boldsymbol{\Sigma}_d\boldsymbol{V}_d^T$). Cette réduction permet d'avoir une matrice dense potentiellement beaucoup plus petite pouvant être utilisée dans d'éventuels calculs de similarité. Cette réduction en facteurs latents permet aussi potentiellement d'avoir une meilleure généralisation des données puisque des utilisateurs ou items similaires, mais pas nécessairement identique dans la base vectorielle originale le seront aussi dans la nouvelle base vectorielles des facteurs latents. Un utilisateur $\boldsymbol{x} \in \mathbb{R}^i$ (ou item $\boldsymbol{y} \in \mathbb{R}^u$) éventuel pourra être transformé dans la nouvelle base par $\boldsymbol{x}' = \boldsymbol{V}_d^T\boldsymbol{x}$ (ou $\boldsymbol{y}' = \boldsymbol{U}_d^T\boldsymbol{y}$).

Par exemple, dans un contexte de recommandation de films, ces facteurs latents obtenus représenteront potentiellement le genre des films et permettront de voir que deux utilisateurs écoutent le même type de films et ce, sans avoir besoin que les mêmes films particuliers aient été écoutés par ces deux utilisateurs.

3.2 Réseaux de neurones

L'utilisation des réseaux neuronaux a connu dans la dernière dizaine d'années un essor incroyable dû à leur performance dans des situations où le nombre de variables est important

et les relations entre celles-ci ne sont pas évidentes. Depuis quelques années, ceux-ci ont été appliqués aux systèmes de recommandations dans la recommandation de produits, mais peu dans la recommandation de personne à personne, ce qui est le but de l'algorithme présenté au chapitre suivant.

3.2.1 Perceptron multicouche (*Multilayer Perceptron*)

Le perceptron multicouche est l'élément de base de la plupart des réseaux neuronaux. Comme son nom l'indique, celui-ci est composé de plusieurs couches suivant toutes le même schéma, soit une transformation linéaire suivi d'une fonction appliquée par élément. Plus concrètement, pour un vecteur de données d'entrées $\boldsymbol{x} \in \mathbb{R}^p$, une matrice de poids $\boldsymbol{W} \in \mathbb{R}^{k \times p}$, un vecteur de biais $\boldsymbol{b} \in \mathbb{R}^k$ et une sortie $\hat{\boldsymbol{y}} \in \mathbb{R}^k$, on aura

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \\ \boldsymbol{y} &= \sigma(\boldsymbol{z}) \end{aligned} \tag{3.4}$$

où σ représente n'importe quelle fonction appliquée par élément. Certaines des plus utilisées sont :
sigmoid :

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

tanh :

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.6}$$

relu :

$$f(x) = \max(0, x) \tag{3.7}$$

Lorsqu'il a y a plus d'une couche, la sortie de l'une est l'entrée de la suivante. Soit $g(\boldsymbol{x})$ le résultat de l'équation 3.4. Le résultat $\hat{\boldsymbol{y}}$ sera donnée par

$$\begin{aligned} \boldsymbol{h}^{(0)} &= g\left(\boldsymbol{x}\right) \\ \boldsymbol{h}^{(l)} &= g\left(\boldsymbol{h}^{(l-1)}\right) \\ \hat{\boldsymbol{y}} &= \boldsymbol{h}^{(L)} \end{aligned} \tag{3.8}$$

où L est le nombre de couches.

Pour un problème de classification, la fonction typiquement utilisée à la dernière couche est

le *softmax* :

$$softmax(\boldsymbol{y}) = \left(\frac{e^{y_i}}{\sum_k e^{y_k}} \right)_i \tag{3.9}$$

Pour ajuster les paramètres de poids et de biais, il faut procéder à l'entraînement du modèle, c'est-à-dire donner des paires de données d'entrée et de sortie justes et faire en sorte que le modèle fasse les meilleures prédictions possibles de celles-ci. La façon typique d'entraîner des modèles de réseaux neuronaux est par la descente de gradient et la rétropropagation (*backpropagation*). Pour ce faire, il faut définir une fonction de coût et calculer le gradient de chacun des paramètres par rapport à celle-ci.

Deux fonctions de coût très utilisées sont la distance quadratique pour les problèmes de régression et l'entropie croisée pour les problèmes de classification.

Pour un problème de régression et des sorties et prédictions données par $y_i, \hat{y}_i \in \mathbb{R}$, la distance quadratique est donnée par :

$$J = \sum_i (\hat{y}_i - y_i)^2 \tag{3.10}$$

Similairement, pour un problème de classification et des sorties et prédictions données par $\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i \in \mathbb{R}^k$, l'entropie croisée est

$$J = -\sum_i \sum_j y_{ij} \log \hat{y}_{ij} \tag{3.11}$$

La rétropropagation consiste à définir les gradients de chacune des couches par rapport aux couches supérieures et d'ainsi pouvoir propager ceux-ci à travers le réseau. Un exemple de ceci peut être vu à la figure 3.1.

Prenons par exemple un problème de régression avec comme fonction de coût la distance quadratique donnée par l'équation 3.10. Avec des données d'entrée $\boldsymbol{x}$ et de sortie y, on définira les différentes couche du MLP par

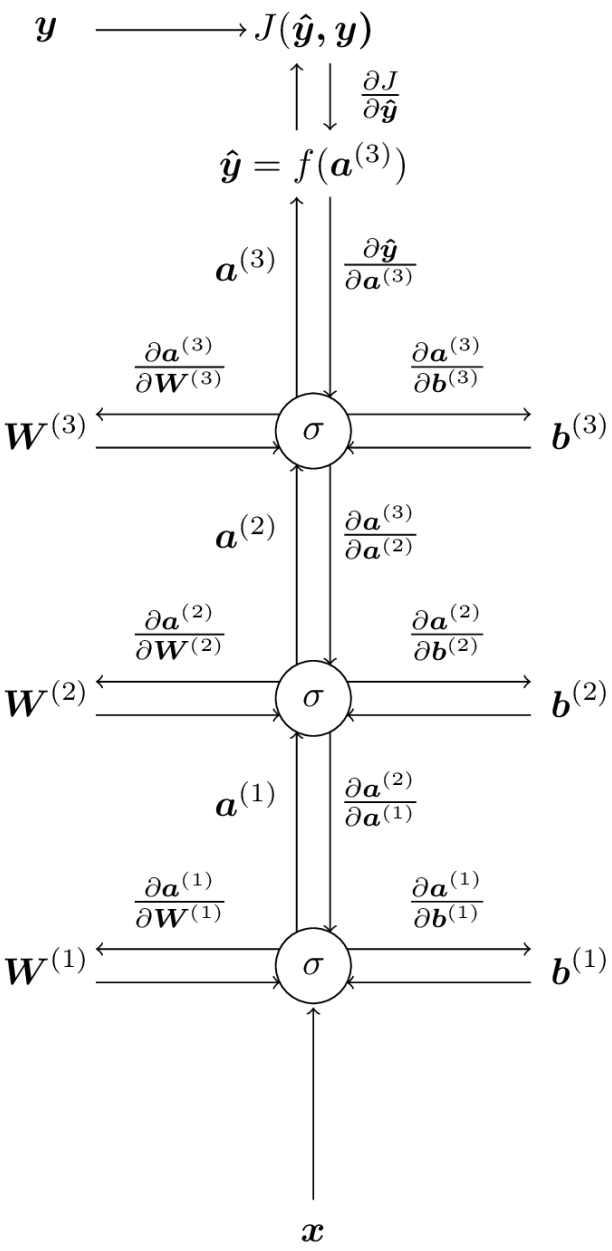


Figure 3.1 Exemple de rétropropagation. On peut voir que la sortie de chaque couche est définie par rapport à la couche inférieure et que le gradient est défini par rapport à la couche supérieure. À chaque couche, σ représente l'opération $\boldsymbol{a}^{(i+1)} = g(\boldsymbol{a}^{(i)}\boldsymbol{W}^{(i+1)} + \boldsymbol{b}^{(i+1)})$ avec g une fonction appliquée par élément et f une fonction dépendant du problème (par exemple softmax pour un problème de classification).

$$
\begin{aligned}
\boldsymbol{z}_1 &= \boldsymbol{W}_1^T \boldsymbol{x} + \boldsymbol{b}_1 \\
\boldsymbol{a}_1 &= \sigma(\boldsymbol{z}_1) \\
\boldsymbol{z}_l &= \boldsymbol{W}_l^T \boldsymbol{a}_{l-1} + \boldsymbol{b}_l \\
\boldsymbol{a}_l &= \sigma(\boldsymbol{z}_l) \\
\hat{y} &= \boldsymbol{w}_L^T \boldsymbol{a}_{L-1} + b_L
\end{aligned}
\tag{3.12}
$$

où σ est la fonction sigmoid définie à l'équation 3.5.

Les gradients seront

$$
\begin{aligned}
\frac{\partial J}{\partial \hat{y}} &= 2(\hat{y} - y) \equiv \delta_L \\
\frac{\partial J}{\partial \boldsymbol{z}_{L-1}} &= (\delta_L \boldsymbol{w}_L) \odot \boldsymbol{a}_{L-1} \odot (\boldsymbol{1} - \boldsymbol{a}_{L-1}) \equiv \boldsymbol{\delta}_{L-1} \\
\frac{\partial J}{\partial \boldsymbol{z}_l} &= (\boldsymbol{W}_{l+1}^T \boldsymbol{\delta}_{l+1}) \odot \boldsymbol{a}_l \odot (\boldsymbol{1} - \boldsymbol{a}_l) \equiv \boldsymbol{\delta}_l \\
\frac{\partial J}{\partial \boldsymbol{w}_L} &= \delta_L \boldsymbol{a}_{L-1} \\
\frac{\partial J}{\partial b_L} &= \delta_L \\
\frac{\partial J}{\partial \boldsymbol{W}_l} &= \boldsymbol{a}_{l-1} \boldsymbol{\delta}_L^T \\
\frac{\partial J}{\partial \boldsymbol{b}_l} &= \boldsymbol{\delta}_l
\end{aligned}
\tag{3.13}
$$

Bien que plusieurs méthodes existent pour optimiser les paramètres en utilisant ces gradients, les approches récentes utilisent une optimisation stochastique, c'est-à-dire utilisant seulement une partie des données à chaque itération pour calculer les gradients et une approximation des moments pour s'assurer que les gradients ne soient pas trop différents d'une itération à l'autre. Un algorithme très populaire et qui est utilisé pour les expériences faites ci-dessous est Adam (Kingma and Ba, 2014), décrit à l'algorithme 1.

Celui-ci utilise des estimés des deux premiers moments des gradients pour avoir des mises à jours plus constantes des paramètres d'une itération à l'autre.

Algorithm 1 Optimisateur Adam

Require: $\alpha > 0$: Taille du pas
Require: $\beta_1, \beta_2 \in [0, 1)$: Taux de déclin exponentiel pour les estimations des moments
Require: $\epsilon > 0$: Paramètre de stabilité numérique
Require: $L(\boldsymbol{\theta})$: Fonction d'objectif à minimiser avec les paramètres $\boldsymbol{\theta}$
Require: $\boldsymbol{\theta}_0$: Paramètre initiaux

$\boldsymbol{m}_0 \leftarrow \mathbf{0}$ ▷ Initialisation du vecteur des 1er moments
$\boldsymbol{v}_0 \leftarrow \mathbf{0}$ ▷ Initialisation du vecteur des 2e moments
$t \leftarrow 0$ ▷ Initialisation de l'intervalle de temps
while θ_t n'a pas convergé **do**
  $t \leftarrow t + 1$
  $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} L_t(\boldsymbol{\theta}_{t-1})$ ▷ obtention des gradients par rapport à l'objectif au temps t
  $\boldsymbol{m}_t \leftarrow \beta_1 \cdot \boldsymbol{m}_{t-1} + (1 - \beta_1) \cdot \boldsymbol{g}_t$ ▷ Mise à jour de l'estimation du 1er moment
  $\boldsymbol{v}_t \leftarrow \beta_2 \cdot \boldsymbol{v}_{t-1} + (1 - \beta_2) \cdot \boldsymbol{g}_t^2$ ▷ Mise à jour de l'estimation du 2e moment
  $\hat{\boldsymbol{m}}_t \leftarrow \frac{\boldsymbol{m}_t}{1-\beta_1^t}$ ▷ Calcul du 1er moment corrigé pour le biais
  $\hat{\boldsymbol{v}}_t \leftarrow \frac{\boldsymbol{v}_t}{1-\beta_2^t}$ ▷ Calcul du 2e moment corrigé pour le biais
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \cdot \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}$ ▷ Mise à jour des paramètres
end while
return θ_t ▷ Paramètres obtenus

3.2.2 Réseaux neuronaux récurrents

Le perceptron multicouche est très utile lorque les entrées sont dans un format constant, mais ne peut s'adapter à des séries de longueur inégales comme la parole ou le texte. Heureusement, les réseaux neuronaux récurrents (*Recurrent Neural Networks*) ou RNN ont été créés pour palier ce problème.

L'idée derrière cette class de modèles est de réutiliser les mêmes paramètres pour chaque élément de la série et d'accumuler une représentation de celle-ci.

L'idée derrière cette classe de modèles est de réutiliser les mêmes paramètres à toutes les étapes.

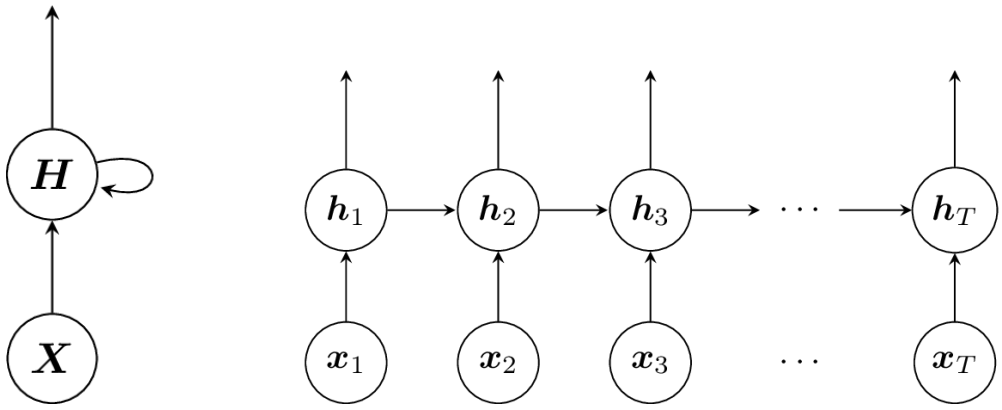
Ceux-ci gardent une mémoire interne et accumulent celle-ci tout au long de la progression sur les données. Ils font ceci en calculant une combinaison linéaire de la mémoire à l'étape précédente et des données de l'étape actuelle. Ceci se traduit par l'équation :

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}) \tag{3.14}$$

où σ représente n'importe quelle fonction appliquée par élément.

Le vecteur $\boldsymbol{h}^{(t)}$ est une représentation de la séquence d'entrées $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}\}$ et pour

une séquence de longueur T, sa représentation sera encodé dans $\boldsymbol{h}^{(T)}$.



(a) Réseau roulé (*rolled network*)

(b) Réseau déroulé (*unrolled network*)

Figure 3.2 Un réseau neuronal récurrent peut être représenté de deux façons, illustrant les deux idées principales de celui-ci. Le réseau roulé (*gauche*) permet de voir qu'il y a une récurrence dans le calcul des représentations $\boldsymbol{H}$ et une réutilisation des paramètres. Le réseau déroulé (*droite*) montre de quelle façon les réprésentations $\boldsymbol{h}_t$ sont accumulées tout au long de la séquence.

Un exemple de ceci peut être vu à la figure 3.2. On peut voir que chaque élément de la séquence $\boldsymbol{x}_t$ est utilisé avec la représentation précédente $\boldsymbol{h}_{t-1}$ pour créer une nouvelle représentation $\boldsymbol{h}_t$. Celle-ci est alors utilisée dans le calcul de la représentation suivante ainsi que comme sortie pour l'élément courant.

Ce type de réseau est souvent utilisé pour des problèmes où la séquence est de taille variable comme le traitement de textes ou sons. Ces réseaux ont typiquement deux utilités : obtenir une représentation à chaque étape et/ou obtenir une représentation finale de toute la séquence. On peut voir à la figure 3.3 un exemple où chaque mot de la phrase est étiqueté alors que pour la figure 3.4, seul la représentation finale est utile pour le classificateur.

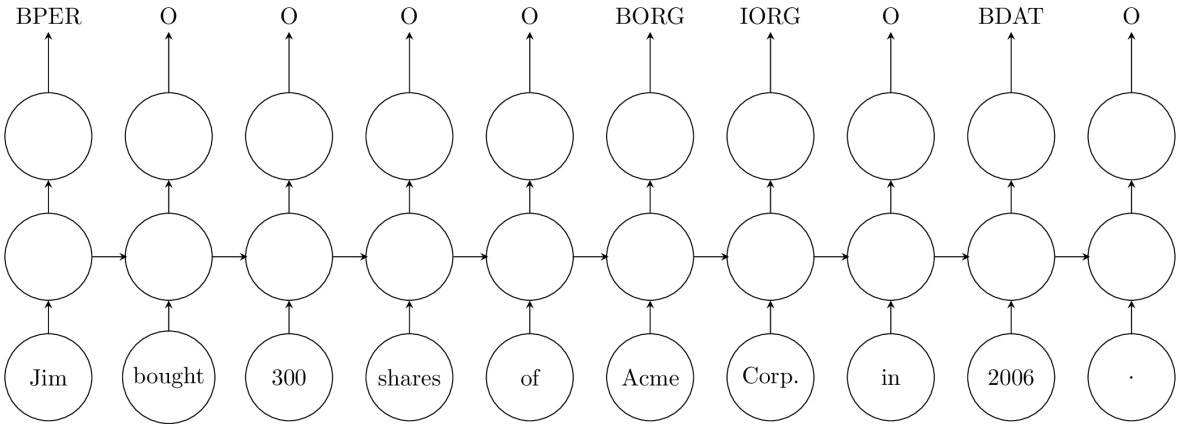


Figure 3.3 Exemple de réseau récurrent utilisé pour trouver des entités comme la personne, l'organisme et la date. La couche du centre est la couche récurrente qui envoit à chaque étape la représentation courante à un classificateur.

De la même façon que pour les perceptrons multicouches, un réseau récurrent peut avoir plusieurs couches et tous les paramètres seront entraînés par rétropropagation.

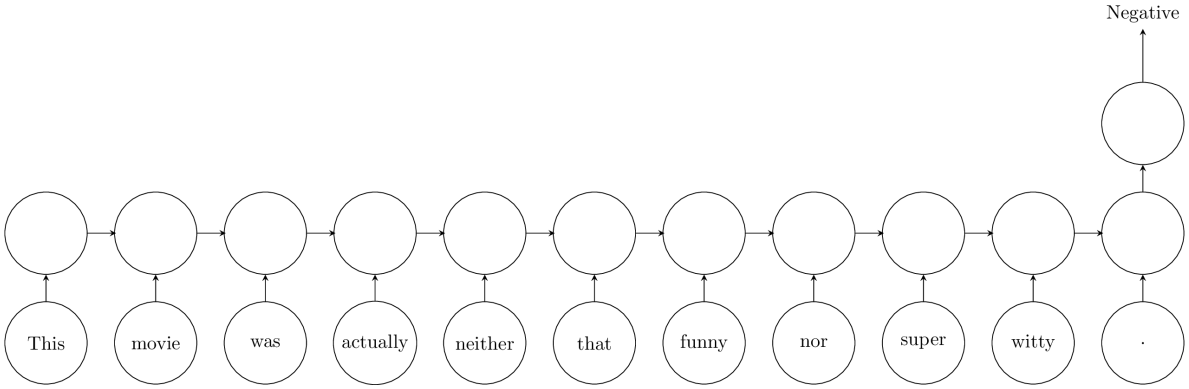


Figure 3.4 Exemple de réseau récurrent utilisé pour déterminer si la phrase est positive ou négative. La représentation finale encode en théorie toutes les nuances de la phrases permettant au classificateur de donner la réponse juste.

CHAPITRE 4 REPRÉSENTATION ET RECOMMANDATION D'UTILISATEURS À L'AIDE DES RÉSEAUX NEURONAUX

4.1 Introduction

L'objectif est de recommander à une personne d'autres utilisateurs partageant la même motivation d'apprentissage ou d'enseignement autour de sujets à maîtriser ou de compétences à acquérir. Les données utilisées proviennent du site `www.e-180.com` (E180, 2018) qui recommande des utilisateurs dans un contexte de transfert de connaissances. Chaque utilisateur peut émettre une offre pour enseigner ou faire une requête pour apprendre, ce qu'on l'on nommera **expertise** dans le reste du document. Chacune de celles-ci comporte trois éléments : un titre, une description et des *tags*. Ces derniers sont des mots-clés entrés par l'utilisateur et dont l'orthographe est corrigée pour éviter la création de *tags* redondants.

Puisque le site ne limite pas le type de sujet et que chaque utilisateur peut avoir autant d'expertises qu'il veut, celui-ci peut avoir des intérêts très diversifiés, rendant sa représentation sous forme vectorielle potentiellement très complexe.

Puisque le site possède à la fois des utilisateurs francophones, anglophones ou bilingues, le modèle doit être utilisable pour plusieurs langues en même temps et devrait aussi être généralisable à d'autres langages qui pourraient potentiellement aussi être utilisés.

Un exemple d'utilisateur peut être vu au tableau 4.1.

Tableau 4.1 Exemple d'utilisateur

Utilisateur	**Expertise**	**Titre français**	**Description française**	**Tags français**
7	1	Apprendre à nager	Je voudrais apprendre à nager.	nager, apprendre
		Titre anglais	**Description anglaise**	**Tags anglais**
		Learning to swim	I would like to learn to swim.	swimming, learning
Utilisateur	**Expertise**	**Titre français**	**Description française**	**Tags français**
7	2	Bâtir des sites webs	Je peux vous montrer comment bâtir des sites web avec HTML et CSS.	HTML, CSS, site web
		Titre anglais	**Description anglaise**	**Tags anglais**
		Building websites	I can show you how to make websites with HTML and CSS.	HTML, CSS, website

Chaque utilisateur a aussi un historique des profils qu'il a visités, des utilisateurs avec qui il a eu une conversation et de ceux qu'il a rencontrés par l'entremise du site.

On peut donc séparer les données en deux : les données d'expertise et les données collaboratives. Des statistiques sur celles-ci sont rapportées à la table 4.2.

Tableau 4.2 Statistiques générales sur les données

Données de contact	
Utilisateurs actifs	17 592
Vues	163 657
Messages	50 204
Rencontres	24 090
Total	237 951
Utilisateurs avec une expertise	
Français	2908
Anglais	6077
Bilingue	331
Total	8654
Nombre d'expertises	
Français	7770
Anglais	9643
Bilingue	664
Total	16 749
Nombre moyen de mots par expertise	
Français	14.834
Anglais	20.044
Mots-clés uniques	12 141
Nombre moyen de mot-clé par expertise	4.704

4.1.1 Représentation des données

Les données d'expertise seront simplifiées de deux façons. Puisque certaines expertises n'ont que le titre, celui-ci sera ajouté au début de la description. Quant aux *tags*, plutôt que de les garder séparés par langages, ils seront rassemblés ensemble. Par exemple, les données du tableau 4.1 transformées se trouvent dans le tableau 4.3.

Tableau 4.3 Exemple d'utilisateur transformé

Utilisateur	**Expertise**	**Description française**	**Tags**
7	1	Apprendre à nager Je voudrais apprendre à nager.	nager, apprendre, swimming, learning
		Description anglaise	
		Learning to swim I would like to learn to swim.	
Utilisateur	**Expertise**	**Description française**	**Tags**
7	2	Bâtir des sites webs Je peux vous montrer comment bâtir des sites web avec html et css.	html, css, site web, websites
		Description anglaise	
		Building websites I can show you how to make websites with html and css.	

Pour ce qui est des données collaboratives, celles-ci seront transformées en une matrice creuse $\boldsymbol{C}$ de la forme suivante : pour chaque combinaison d'utilisateurs i et j, l'élément $C_{i,j}$ de la matrice sera calculé de la façon suivante

$$C_{i,j} = \begin{cases} 0 & \text{s'il n'y a aucune activité entre les deux utilisateurs} \\ +1 & \text{si l'utilisateur } i \text{ a consulté le profil de l'utilisateur } j \\ +2 & \text{s'il y a eu une conversation entre les deux utilisateurs} \\ +4 & \text{s'il y a eu une rencontre entre les deux utilisateurs} \end{cases} \tag{4.1}$$

Les éléments de $\boldsymbol{C}$ seront donc entre 0 et 7.

Un exemple de cette matrice $\boldsymbol{C}$ est

$$\begin{pmatrix} 0 & 1 & 7 & 5 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & \cdots & 1 \\ 6 & 0 & 1 & 0 & 5 & 0 & \cdots & 3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 3 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

4.2 Modèle de référence

Pour évaluer la validité d'un nouveau modèle, nous le comparons au modèle actuel le plus performant. Dans le cas des données utilisées dans ce document, ce modèle a été présenté dans Spaeth and Desmarais (2013).

Celui-ci combine l'analyse sémantique latente (*Latent Semantic Analysis*) ou LSA (basée sur la décomposition SVD, section 3.1.3) avec des filtres collaboratifs. Ce modèle a été réimplémenté le plus fidèlement possible dans le cadre de ce document.

L'analyse sémantique latente se fait en deux étapes : la conversion de chacune des descriptions en vecteurs TF-IDF (*Term Frequency – Inverse Document Frequency*) et la réduction de ceux-ci par la décomposition en vecteurs de support.

4.2.1 *Term Frequency – Inverse Document Frequency* (TF-IDF)

Pour obtenir les vecteurs TF-IDF, deux informations sont nécessaires : la fréquence des mots pour chaque description (TF) et la fréquence de ces mots dans chacune des descriptions (IDF).

Pour chaque description, la fréquence des mots est créant un vecteur où chaque élément représente la quantité de fois où le mot correspondant apparaît dans cette description. Par exemple, pour le vocabulaire

$$\begin{pmatrix} \text{j'} & \text{ai} & \text{un} & \text{peu} & \text{aimer} & \text{le} & \text{peur} & \text{de} & \text{nager} & , & ? & \text{écrit} & \dots & \text{mais} & \text{d'} & \text{aide} & \text{bleu} & \text{vite} & \text{arriverai} & \text{à} & . & \text{sauter} & \text{avion} \end{pmatrix}$$

la phrase «J'ai un peu peur de nager, mais avec un peu d'aide j'arriverai à nager.» donnerait le vecteur suivant :

$$\begin{pmatrix} 2 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 2 & 1 & 0 & 0 & \cdots & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

En transformant chacune des descriptions de cette façon et en les concaténant l'une par-dessus l'autre, on obtiendra une matrice documents-termes de taille $|D| \times |T|$ où D représente l'ensemble des descriptions et T le vocabulaire.

Le problème avec cette représentation est que les mots peu communs et plus représentatifs des différences auront un petit poids comparé aux mots apparaissant de façon très commune dans la plupart des descriptions.

Une façon de résoudre ce problème est de moduler ces poids par l'inverse de la fréquence dans les documents. Pour le vocabulaire de termes T et l'ensemble de descriptions D définis précédemment, ce vecteur de fréquences est calculé par

$$\boldsymbol{idf} = \left(\log \left(\frac{|D|}{|\{d \in D : t \in d\}|} \right) \right)_{t \in T}$$

Pour un vocabulaire qui n'est pas déterminé par les données, la formule suivante peut être utilisée pour éviter la division par 0 lorsqu'un mot n'apparaît dans aucun des documents.

$$\boldsymbol{idf} = \left(\log \left(\frac{|D|}{1 + |\{d \in D : t \in d\}|} \right) \right)_{t \in T}$$

La matrice finale $\boldsymbol{TFIDF}$ est obtenue en multipliant chaque ligne de la matrice $\boldsymbol{TF}$ par ce vecteur $\boldsymbol{idf}$.

4.2.2 Décomposition en valeurs singulières (SVD)

Maintenant que les expertises ont été transformées par TF-IDF, chacune est dans un format standard et deux expertises avec des mots en commun seront représentées par des vecteurs similaires. Cependant, un problème survient lorsque deux expertises représentent des sujets similaires, mais avec des mots différents. Par exemple, une expertise pourrait être :

Expert en alimentation canine

et une seconde être :

Je voudrais savoir quelle nourriture donner à mon chien

Bien que ces deux expertises concernent le même sujet, elles ne contiennent aucun mot en commun. Une façon de régler ce problème est de transformer la matrice documents-termes *TF-IDF* par SVD (section 3.1.3) de façon à représenter celle-ci dans l'espace engendrée par les k premiers vecteurs propres des colonnes. Le but de ceci est de capturer les sujets généraux des expertises plutôt que leurs mots individuels. Cette décomposition résultera en des vecteurs d'expertise $\boldsymbol{e}$.

4.2.3 Comparaison des expertises

Une fois toutes les expertises transformées par TF-IDF et puis par SVD, celles-ci peuvent être comparées. Pour chaque paire d'expertises $\boldsymbol{e}_i$ et $\boldsymbol{e}_j$, leur similarité est calculée par la formule du cosinus :

$$s(\boldsymbol{e}_i, \boldsymbol{e}_j) = \frac{\boldsymbol{e}_i^T \boldsymbol{e}_j}{\|\boldsymbol{e}_i\| \|\boldsymbol{e}_j\|}$$

Puisque chaque utilisateur peut avoir plusieurs expertises, la similarité entre deux utilisateurs est maintenant représentée par une matrice. Pour les expertises de l'utilisateur i, $\boldsymbol{e}_{ik}, k = 1, \ldots, n_i$ et celles de l'utilisateur j, $\boldsymbol{e}_{jl}, l = 1, \ldots, n_j$, la matrice de similarité sera

$$\boldsymbol{S_{ij}} = \begin{bmatrix} s(\boldsymbol{e}_{i1}, \boldsymbol{e}_{j1}) & s(\boldsymbol{e}_{i1}, \boldsymbol{e}_{j2}) & \ldots & s(\boldsymbol{e}_{i1}, \boldsymbol{e}_{jn_j}) \\ s(\boldsymbol{e}_{i2}, \boldsymbol{e}_{j1}) & s(\boldsymbol{e}_{i2}, \boldsymbol{e}_{j2}) & \ldots & s(\boldsymbol{e}_{i2}, \boldsymbol{e}_{jn_j}) \\ \vdots & \vdots & \ddots & \vdots \\ s(\boldsymbol{e}_{in_i}, \boldsymbol{e}_{j1}) & s(\boldsymbol{e}_{in_i}, \boldsymbol{e}_{j2}) & \ldots & s(\boldsymbol{e}_{in_i}, \boldsymbol{e}_{jn_j}) \end{bmatrix}$$

Pour faire la recommandation d'utilisateurs, il faut pouvoir comparer des scalaires entre eux et non des matrices. Pour ce faire, deux méthodes d'agrégation sont utilisées : par moyenne et par maximum. Pour la première, la similarité entre deux utilisateurs est la moyenne des similarités de leurs expertises, alors que pour la seconde, le maximum de similarité est retenu. expertises.

L'intuition pour adopter la moyenne est que deux utilisateurs seront intéressés à communiquer s'ils ont plusieurs intérêts en commun alors que le maximum cible de façon plus précise une connexion basée sur un seul intérêt commun.

4.2.4 Combinaison avec filtres collaboratifs

Pour calculer les similarités entre utilisateurs grâce à des filtres collaboratifs, la matrice $\boldsymbol{C}$ décrite à l'équation 4.1 est utilisée.

Celle-ci est réduite par SVD sur les colonnes et la similarité du cosinus est ensuite calculée entre chacune des lignes représentant la projection réduite de chacun des utilisateurs.

4.2.5 Recommandations

Lors de l'évaluation et de l'utilisation du modèle, des recommandations doivent être faites. Par souci de simplicité, les recommandations faites sont les k utilisateurs avec les similarités

les plus hautes. Ce k est un hyperparamètre contrôlant le compromis entre la précision et l'étendue des recommandations.

Pour s'assurer de n'évaluer le modèle que sur de nouvelles données pour l'ensemble de test, les similarités pour les utilisateurs déjà contactés (par un message ou une rencontre) sont ignorées lors du choix des k meilleures recommandations.

4.3 Modèle

Le modèle de référence peut seulement utiliser les fréquences de mots pour établir la similarité entre deux utilisateurs sans pouvoir utiliser les subtilités de langages permettant potentiellement de faire des recommandations plus pertinentes et plus surprenantes. Les réseaux de neurones ont montrés qu'ils étaient capables d'extraire des informations complexes à partir du texte en utilisant notamment des *embeddings* de mots (décrits à la section 4.3.1) entraînés sur d'immenses corpus.

Le modèle décrit ci-dessous profite de cette représentation plus complexe pour tenter de faire des recommandations plus diversifiées et surprenantes que les modèles typiquement utilisés.

Le modèle vu à la figure 4.1 est décomposé en plusieurs étapes créant une hiérarchie dont le résultat final est un vecteur représentant un utilisateur particulier. Chacune de ces étapes est décrite ci-dessous.

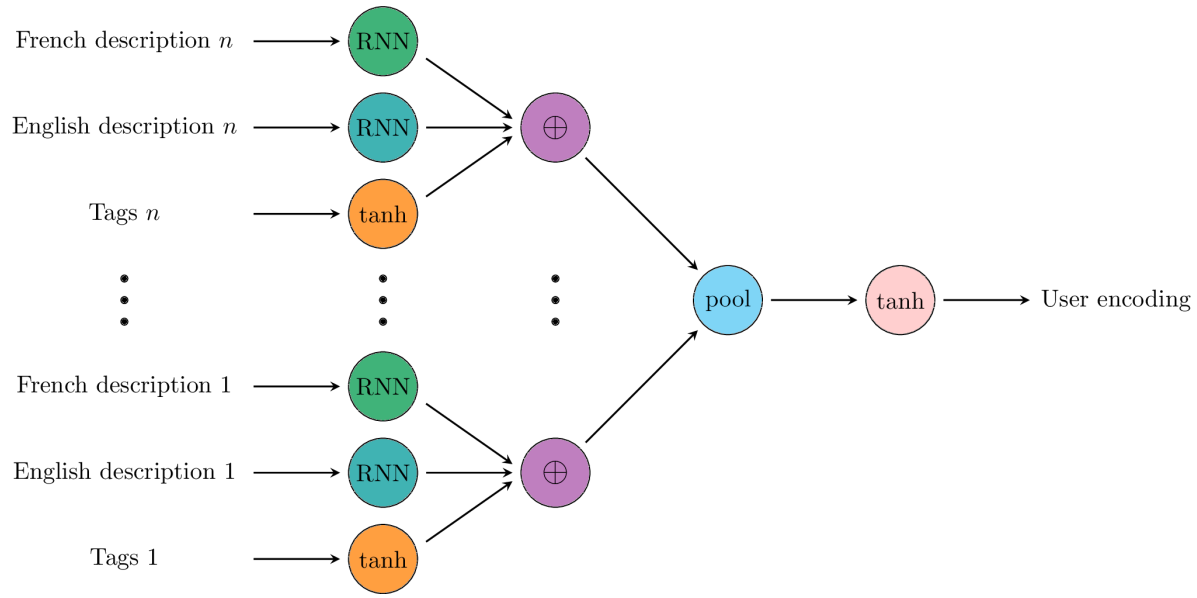


Figure 4.1 Modèle créant une représentation vectorielle pour un utilisateur donné à partir des descriptions et *tags* de ses expertises. On a que *tanh* représente une transformation linéaire suivie de la fonction *tanh* appliquée par élément et que $\oplus$ est l'opération de concaténation.

4.3.1 Encodage des descriptions

L'encodage des descriptions se fait en deux étapes : la conversion de chaque mot en *embedding* puis une transformation de ceux-ci à l'aide d'un réseau neuronal récurrent.

Conversion en *embeddings*

L'*embedding* d'un mot est un vecteur de taille fixe qui en encode l'information sémantique et syntaxique. Ces *embeddings* sont normalement entrainés en se basant sur le postulat que les mots similaires apparaissent ensemble ou dans des contextes similaires, c'est-à-dire entourés des mêmes expressions. Ceux-ci peuvent être entraînés par descente de gradient à l'intérieur du modèle même où ils sont utilisés, mais ceux-ci seront plus représentatifs s'ils sont entrainés précédemment sur d'immenses corpus de mots à l'aide de techniques d'apprentissage non supervisés telles que *word2vec* (Mikolov et al., 2013) ou *GloVe* (Pennington et al., 2014).

Dans le cas de travail, les *embeddings* anglais ont été pris de *GloVe* et ceux français de Fauconnier (2016).

Chaque description i sera transformée en une série d'*embedding* $\boldsymbol{e}_{it} \in \mathbb{R}^{n_e}$ avec $t \in 1, \ldots, T$ indiquant la position de ce mot dans la phrase et n_e la taille de chaque *embedding*.

Ces *embeddings* sont par la suite accumulés par un réseau de neurones récurrent ou *Recurrent Neural Network* (*RNN*), un exemple duquel peut être vu à la figure 4.2. Pour une expertise donnée i, on aura :

$$\begin{aligned} \boldsymbol{h}_0 &= \boldsymbol{0} \\ \boldsymbol{h}_t &= \tanh(\boldsymbol{W}_d^T \boldsymbol{e}_{it} + \boldsymbol{U}_d^T \boldsymbol{h}_{t-1} + \boldsymbol{b}_d) \\ \hat{\boldsymbol{y}}_{id} &= \boldsymbol{h}_T \end{aligned} \tag{4.2}$$

avec $\boldsymbol{W}_d \in \mathbb{R}^{n_e \times n_h}$, $\boldsymbol{U}_d \in \mathbb{R}^{n_h \times n_h}$ et $\boldsymbol{b}_d \in \mathbb{R}^{n_h}$.

Le sens sémantique et syntaxique de la description est encodé dans ce dernier vecteur $\hat{\boldsymbol{y}}_{id}$. Puisque chaque expertise possède une description en français et en anglais, deux réseaux récurrents seront créés et retourneront chacun un vecteur qu'on nommera $\boldsymbol{f}_i$ et $\boldsymbol{a}_i$ pour les deux langages.

4.3.2 Encodage des *tags*

Puisque les *tags* sont représentés par des indices dans les données, on ne peut pas les transformer en *embeddings* comme pour les descriptions. On les représentera donc par un vecteur

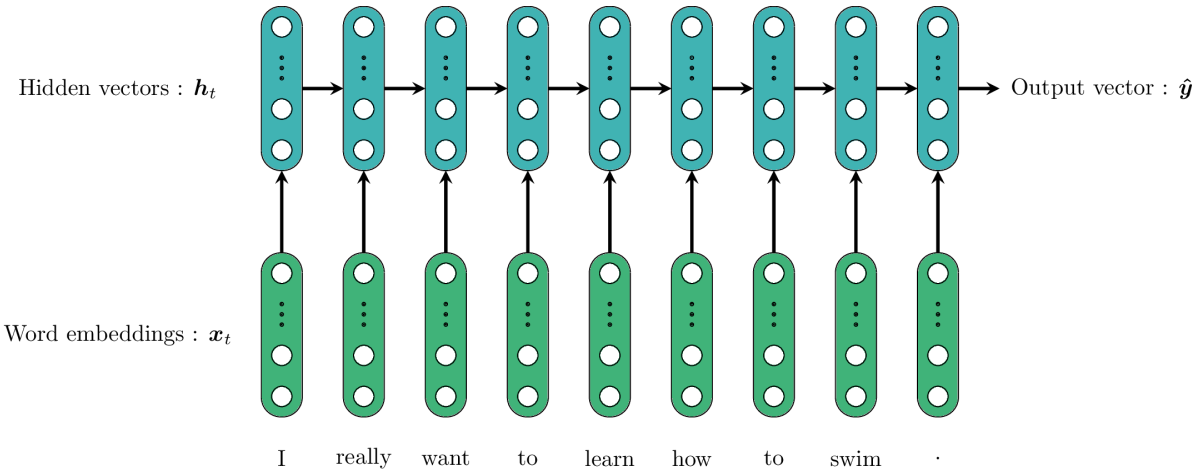


Figure 4.2 Exemple de réseau récurrent

de taille n_t où chaque élément est binaire et indique si le *tag* correspondant est présent ou pas dans l'expertise.

Les *tags* d'une expertise i, $\boldsymbol{x}_i$, seront par la suite transformés par :

$$\boldsymbol{t}_i = \tanh(\boldsymbol{W}_t^T \boldsymbol{x}_i + \boldsymbol{b}_t) \tag{4.3}$$

où la fonction

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.4}$$

est appliqué par élément et $\boldsymbol{W}_t \in \mathbb{R}^{n_t \times 2n_h}$, $\boldsymbol{b} \in \mathbb{R}^{2n_h}$.

4.3.3 Encodage de chaque utilisateur

Chaque expertise est donc composée d'un vecteur pour les *tags* et d'un vecteur pour la description de chacun des langages ; dans ce cas-ci, français et anglais. Ces vecteurs sont concaténés pour former un vecteur représentant l'expertise.

Un utilisateur peut posséder plusieurs expertises et celles-ci doivent être combinées en un seul vecteur le représentant. Ceci se fait par la technique du *pooling* par moyenne ou maximum, c'est-à-dire en créant un nouveau vecteur dont chaque élément est la moyenne ou le maximum de l'élément correspondant de chacun des vecteurs d'expertise. Un exemple de ceci peut être vu dans la figure 4.3.

Finalement, ce vecteur $\boldsymbol{p} \in \mathbb{R}^{4n_h}$ est transformé par

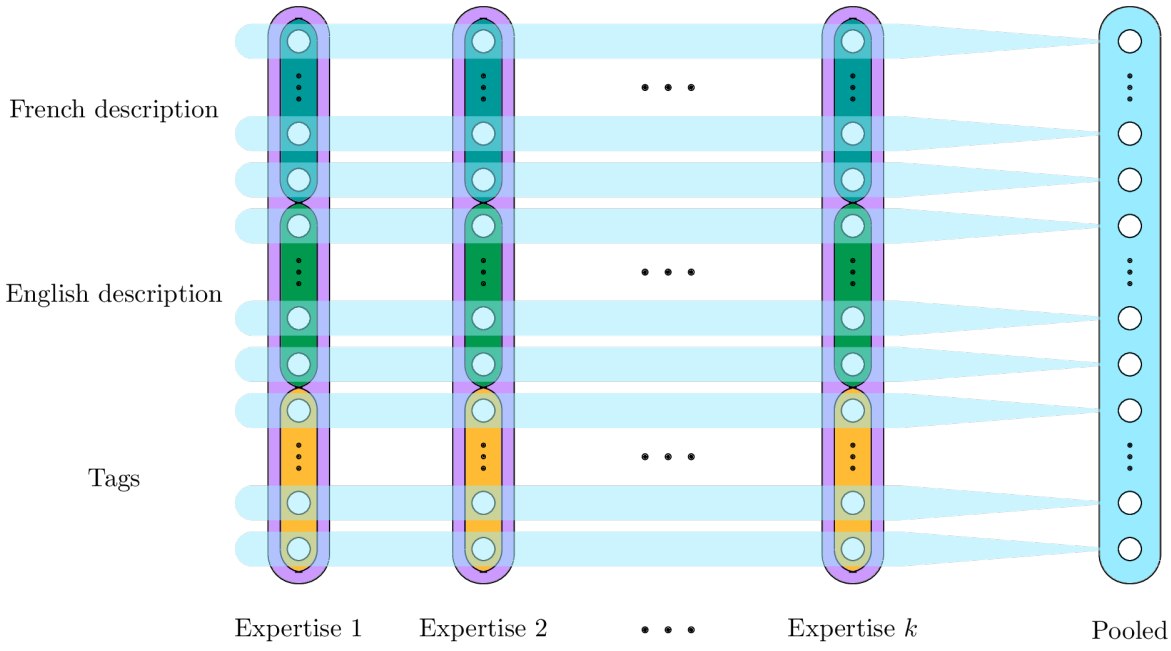


Figure 4.3 Exemple de pooling

$$\boldsymbol{u} = \tanh(\boldsymbol{W}_u^T \boldsymbol{p} + \boldsymbol{b}_u) \tag{4.5}$$

où $\boldsymbol{W}_u \in \mathbb{R}^{4n_h \times n_u}$, $\boldsymbol{b} \in \mathbb{R}^{n_u}$.

4.3.4 Entraînement

Pour entraîner le modèle, il faut définir une fonction de perte à optimiser. Pour ce faire, les données collaboratives décrites à la section 4.1.1 sont utilisées. Chaque élément de cette matrice représente une magnitude de rapprochement voulue entre les utilisateurs correspondant à la ligne et colonne. Une rencontre avec un poids de 4 indique un rapport plus important entre les deux individus que le visionnement du profil de l'un ou l'autre qui n'a qu'un poids de 1.

L'entraînement se fait de façon itérative en prenant à chaque itération un échantillon de taille b des valeurs non nulles de cette matrice collaborative $\boldsymbol{C}$. Ceci donnera un vecteur de valeurs $c_i, i = 1, \ldots, b$. Ensuite des vecteurs $\boldsymbol{u}_i$ et $\boldsymbol{v}_i$ sont calculés par le modèle ci-dessus pour les utilisateurs correspondants à la ligne et la colonne de l'élément c_i dans la matrice $\boldsymbol{C}$.

La fonction de perte est composée de deux parties. La première est une mesure de la distance entre ces deux vecteurs $\boldsymbol{u}_i$ et $\boldsymbol{v}_i$ pesée par le rapprochement voulu c_i. Plus le rapprochement est important, plus cette distance aura un poids important pour la perte. Celle-ci est donnée

par :

$$\begin{aligned} J' &= \frac{1}{b}\sum_{i=1}^{b}\sum_{j=1}^{n_u} c_i(u_{ij} - v_{ij})^2 \\ &= \frac{1}{b}\sum_{i=1}^{b} c_i(\boldsymbol{u}_i - \boldsymbol{v}_i)^T(\boldsymbol{u}_i - \boldsymbol{v}_i) \end{aligned} \tag{4.6}$$

Le problème d'utiliser seulement cette équation est qu'au point de convergence, le modèle fera en sorte que tous les vecteurs retournés soient identiques, peu importe l'entrée. Pour pallier à ce problème, cette fonction est régularisée en ajoutant l'inverse de la distance entre chaque vecteur $\boldsymbol{u}_i$ et la moyenne de tous les vecteurs $\boldsymbol{v}_j, j = 1, \ldots, b$ et vice-versa, proportionnée par un hyperparamètre α. Cette régularisation est donnée par :

$$\begin{aligned} \Lambda &= -\frac{\alpha}{b}\left(\sum_{i=1}^{b}(\boldsymbol{u}_i - \bar{\boldsymbol{v}})^T(\boldsymbol{u}_i - \bar{\boldsymbol{v}}) + (\boldsymbol{v}_i - \bar{\boldsymbol{u}})^T(\boldsymbol{v}_i - \bar{\boldsymbol{u}})\right) \\ \bar{\boldsymbol{u}} &= \left(\frac{1}{b}\sum_{i=1}^{b} u_{ij}\right)_{j=1,\ldots,n_u} \\ \bar{\boldsymbol{v}} &= \left(\frac{1}{b}\sum_{i=1}^{b} v_{ij}\right)_{j=1,\ldots,n_u} \end{aligned} \tag{4.7}$$

La fonction de perte finale sera donc une combinaison des deux équations précédentes :

$$\begin{aligned} J &= J' + \Lambda \\ &= \frac{1}{b}\sum_{i=1}^{b} c_i(\boldsymbol{u}_i - \boldsymbol{v}_i)^T(\boldsymbol{u}_i - \boldsymbol{v}_i) - \frac{\alpha}{b}\left(\sum_{i=1}^{b}(\boldsymbol{u}_i - \bar{\boldsymbol{v}})^T(\boldsymbol{u}_i - \bar{\boldsymbol{v}}) + (\boldsymbol{v}_i - \bar{\boldsymbol{u}})^T(\boldsymbol{v}_i - \bar{\boldsymbol{u}})\right) \end{aligned} \tag{4.8}$$

On peut voir que cette fonction de perte est similaire à l'échantillonnage négatif de Mikolov et al. (2013) ou l'analyse discriminante linéaire (LDA) qui détermine la meilleure classe en comparant la vraisemblance de celle-ci à celle des autres classes (Friedman et al., 2001, chap 4.3)

Une fois cette fonction définie, il est possible de calculer les gradients pour tous les paramètres du modèles par rétropropagation grâce à une librairie comme Tensorflow. À chaque itération, chacun des paramètres aura les $2b$ gradients provenant de chacun des vecteurs $\boldsymbol{u}_i$ et $\boldsymbol{v}_i$. Le

gradient utilisé pour modifier le paramètre sera simplement la moyenne de ceux-ci.

L'optimisateur utilisé est Adam (Kingma and Ba, 2014) décrit à l'algorithme 1 avec les paramètres par défaut de l'implémentation de Tensorflow, soit 0.001 pour le taux d'apprentissage, 0.9 et 0.999 pour β_1 et β_2, et 10^{-8} pour ϵ.

Lors de l'entraînement, une partie de l'ensemble d'entraînement a été utilisé comme ensemble de validation pour déterminer les valeurs optimales pour n_e, n_h et n_u. Pour n_e, les valeurs essayées ont été 200, 300 et 500, et pour n_h et n_u, 10, 50, 100, 300 et 500. Les valeurs optimales ont été trouvées en prenant des combinaisons aléatoires parmi ces valeurs 20 fois et le résultat optimale a été d'utiliser 50 pour n_h et n_u et 200 pour n_e.

Pour trouver la valeur de b, moins d'expériences ont pu être faites puisque la capacité de mémoire de l'ordinateur était limitée et une valeur de 8 a donc été prise. Celle-ci donnait de meilleurs résultats que des valeurs plus petites et il n'est pas déraisonnable de penser que de plus grandes valeurs mèneraient à de meilleurs résultats.

Le choix de $\alpha = 0.5$ a été utilisé pour permettre de comparer le modèle à un modèle de base et a semblé bien fonctionner. En pratique, il est possible qu'une valeur différente mène à de meilleurs résultats.

4.3.5 Recommandation

Une fois le modèle entraîné, il faut pouvoir l'utiliser pour faire des prédictions. Pour ce faire, la similarité du cosinus décrite à l'équation 3.3 est utilisée de la même façon que pour le modèle de référence décrit à la section 4.2.5. Pour chaque utilisateur pour lequel le modèle fait des recommandations, cette similarité est calculée avec tous les autres utilisateurs du système.

Cette approche est très grossière et peut être améliorée pour obtenir de meilleurs résultats ou augmenter la vitesse de l'algorithme dans son ensemble, mais celle-ci a permis de facilement comparer les algorithmes entre eux.

La combinaison avec les résultats des filtres collaboratifs a été faite de façon identique à celle décrite à la section 4.2.4.

4.4 Expériences

Pour l'évaluation des modèles, les données ont été séparées en deux ensembles. Le premier contient les données de 2012 à 2015 et est utilisé pour entraîner les modèles. Le second contient les données du 1er janvier 2016 au 1er juin 2016. Celui-ci permet de tester le modèle

pour trois métriques : la précision, le rappel et le score F1 qui est une combinaison des deux précédentes.

Puisqu'un utilisateur n'aura vu qu'une minorité d'autres profils, il est peu probable que celui-ci ait pu voir tous les profils qui pourraient l'intéresser. Par conséquent, une recommandation sera ignorée lors du test si celle-ci est pour un profil non visionné par l'utilisateur. De plus, un message ou une rencontre auront le même poids et la présence de l'un ou l'autre sera simplement appelée *contact* et représentera un choix positif, c'est-à-dire que la recommandation sera considérée bonne.

On aura ainsi qu'une recommandation juste correspond à la fois à un contact et à un visionnement de profil. Tandis qu'une fausse recommandation correspond à un visionnement de profil non suivi d'un contact.

De la matrice de données collaboratives $\boldsymbol{C}$ définie précédemment, on extraira deux matrices. La première, $\boldsymbol{V}$, est une matrice binaire où chaque élément indique s'il y a un visionnement de profil de l'utilisateur de la colonne par l'utilisateur de la ligne. La seconde, $\boldsymbol{M}$, est aussi une matrice binaire, représentant un contact entre les utilisateurs de ligne et colonne ou pas. Dans les données, certains contacts ne sont pas accompagnés par un visionnement de profil ce qui peut être dû à plusieurs raisons, comme l'envoi d'un message après avoir simplement vu le titre de l'offre ou la demande dans un onglet. Puisqu'il est peu probable qu'un utilisateur ait eu une conversation ou une rencontre avec un sans aucune information sur lui, un élément de $\boldsymbol{V}$ sera positif si l'élément correspondant de $\boldsymbol{M}$ l'est aussi. De plus, r_{ij} représentera l'indice de colonne de la $j$ème recommandation pour l'utilisateur de la $i$ème ligne dans les matrices $\boldsymbol{V}$ et $\boldsymbol{M}$. Par exemple, $M_{ir_{ij}}$ indiquera s'il y a eu un contact entre l'utilisateur i et la $j$ème recommandation pour celui-ci.

La précision est calculée comme le nombre de recommandations justes sur le nombre de recommandations totales, ce qui, dans le cas présent, est :

$$précision = \frac{\sum_i \sum_j M_{ir_{ij}}}{\sum_i \sum_j V_{ir_{ij}}} \tag{4.9}$$

Le rappel, quant à lui, est défini comme le nombre de recommandations justes sur le nombre total de valeurs positives, c'est-à-dire :

$$rappel = \frac{\sum_i \sum_j M_{ir_{ij}}}{\sum_i \sum_j M_{ij}} \tag{4.10}$$

Finalement, le score F1 est défini comme :

$$F1 = 2 \cdot \frac{précision \cdot rappel}{précision + rappel} \tag{4.11}$$

Ces trois métriques seront calculées pour le modèle de base avec un *pooling* de maximum ou de moyenne et pour le modèle de réseaux neuronaux avec les mêmes *pooling*. Le nombre de recommandations ayant une grande influence sur le résultat de ces métriques, celles-ci seront aussi calculées avec les nombres de recommandations $[1, 2, \ldots, 10, 20, \ldots, 100, 150, \ldots, 500]$.

Les modèles précédents seront aussi comparés à un modèle de filtres collaboratifs utilisateur-utilisateur. La matrice $\boldsymbol{C}$ précédente sera réduite par SVD sur les colonnes et une similarité du cosinus calculée entre tous les utilisateurs.

Similairement à Spaeth and Desmarais (2013), les similarités calculées par les modèles LSA et neuronaux seront combinées à celles calculées par les filtres collaboratifs, mais selon la formule suivante :

$$\boldsymbol{S} = (1 - \beta) \cdot \boldsymbol{S}_{CF} + \beta \cdot \boldsymbol{S}_{E} \tag{4.12}$$

où $\boldsymbol{S}_{CF}$ représente la similarité calculée par le filtre collaboratif et $\boldsymbol{S}_{E}$ représente celle calculée par les différentes méthodes utilisant les expertises.

Bien que le paramètre β peut avoir une influence sur la qualité des recommandations, une valeur de 0.5 a été utilisée pour les expériences.

4.5 Résultats

Les courbes de précision-rappel et le score F1 ont été calculées pour le modèle de référence (*LSA*) et le nouveau modèle introduit ici (*Neural*) en utilisant les deux techniques d'agrégation présentées ci-dessus, soit le maximum et la moyenne.

Ces modèles ont aussi été combinés avec une recommandation par filtres collaboratifs. Cette combinaison fonctionne de la même façon pour les modèles neuronaux que celle pour le modèle de référence, décrite à la section 4.2.4.

Les courbes de la figure 4.4 montrent que, par eux-mêmes, les modèles LSA ont une meilleure précision lorsque peu de recommandations sont faites (rappel faible), mais que celle-ci diminue rapidement avec l'augmentation de leur nombre. Les modèles neuronaux, quant à eux, n'ont pas la même précision lorsque peu de recommandations sont faites ($\approx 23\%$ par rapport à $\approx 27\%$), mais celle-ci diminue bien moins rapidement lorsque le nombre de recommandations augmente, passant seulement de $\approx 23\%$ à $\approx 19\%$ avant d'ensuite diminuer de façon similaire aux modèles LSA.

On remarque que la performance des modèles neuronaux est semblable à celle des filtres collaboratifs, ce qui est intuitivement logique, puisque ces données collaboratives ont été utilisées pour l'entraînement des réseaux neuronaux.

La combinaison des modèles LSA et neuronaux avec les filtres collaboratifs présentée dans le deuxième graphe de la figure 4.4 montre que les deux modèles en tirent profit, mais que l'avantage est plus grand pour les modèles LSA. Cependant, le fait que les modèles neuronaux profitent aussi de cette combinaison montre qu'ils ont réussi à créer des représentations utiles des utilisateurs grâce aux descriptions d'expertise et non pas seulement à répliquer les données collaboratives utilisées pour les filtres collaboratifs. Bien que la différence soit moins grande, les modèles LSA restent en général plus précis lorsque le nombre de recommandations est petit et les modèles neuronaux prennent le dessus lorsque ce nombre augmente.

La table 4.4 montre que les modèles neuronaux performent en général mieux que les modèles LSA lorsque les filtres collaboratifs ne sont pas ajoutés sur la base du score F1. La combinaison avec les filtres collaboratifs augmente les résultats de tous les modèles, mais de façon plus drastique pour les modèles LSA, augmentant les scores pour peu de recommandations.

Les graphiques de précision-rappel et les scores F1 montrent que la méthode d'agrégation ne semble pas avoir un grand impact sur la performance des modèles.

Les résultats supérieurs obtenus par la combinaison de modèles neuronaux avec des filtres collaboratifs comparés à chacun de ces modèles utilisés seuls montrent que les réseaux neuronaux apprennent des représentations différentes. Bien que ceux-ci aient des résultats similaires lorsque utilisés seuls, mais que combinés, le score augmente drastiquement montre que les réseaux neuronaux sont capables d'utiliser l'information donnée par les données collaboratives pour créer des représentations utiles et plus complètes des données implicites données par les utilisateurs. Plus d'expériences doivent être faites pour vérifier d'où vient cette information et si elle est effectivement améliorée par les données collaboratives ou simplement par la capacité importante des réseaux neuronaux et par les *embeddings* entraînés sur de larges vocabulaires.

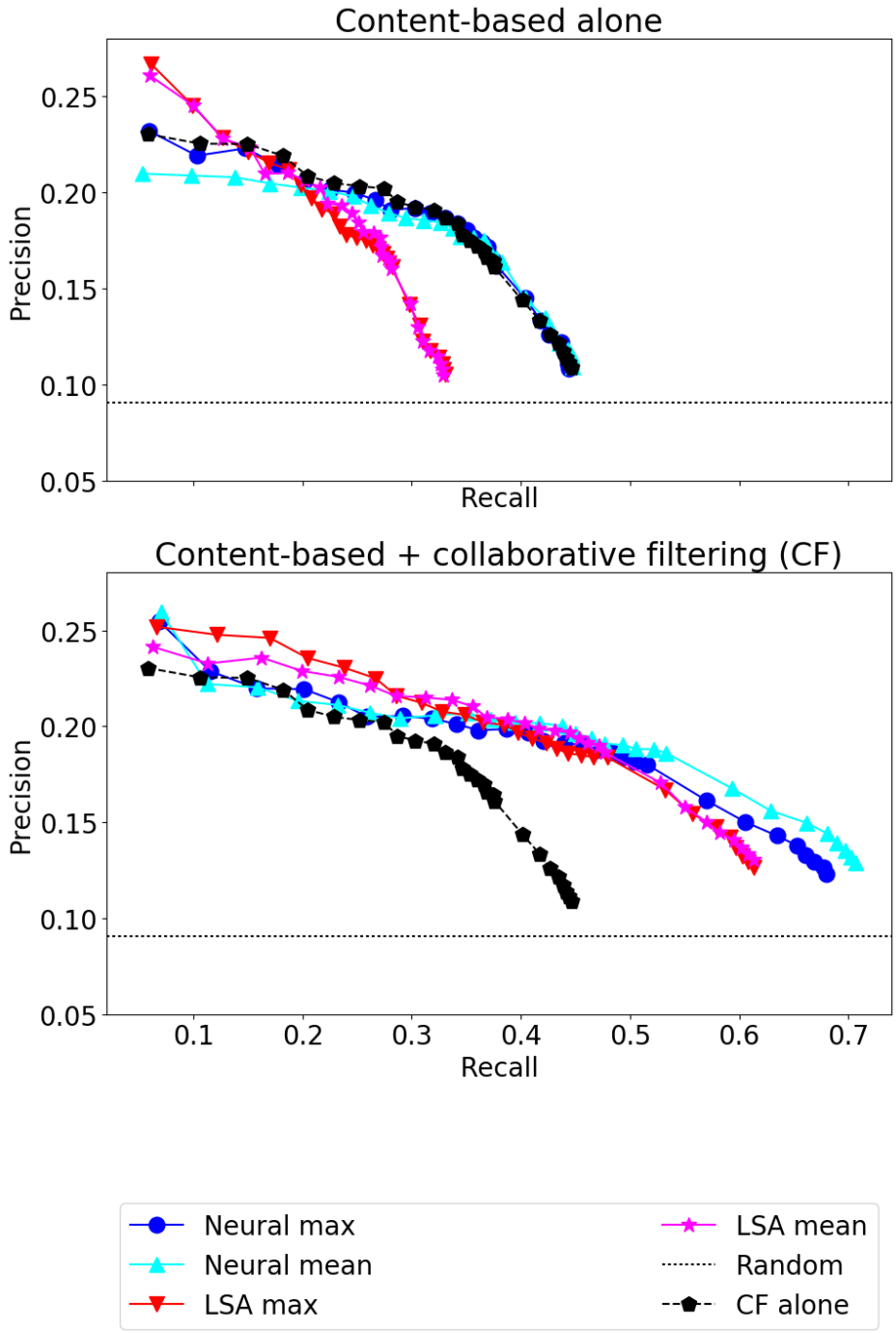


Figure 4.4 Courbes de précision-rappel des modèles à base de réseaux neuronaux et LSA. La figure du haut représente les modèles utilisés seuls alors que celle-ci du bas montre les résultats avec la combinaison avec les filtres collaboratifs. La ligne représentant les résultats pour des recommandations aléatoires est obtenue en recommandant de façon aléatoire parmi les profils vus. En pratique, ceci se traduit par le nombre total de messages ou rencontres divisés par le nombre total de vu de profil dans l'ensemble de test.

Tableau 4.4 Scores F1 avec différents nombres de recommandations. Les nombres en gras indiquent le meilleur score.

	@1	**@5**	**@10**	**@50**	**@100**
Content-based alone					
Neural max	0.094	**0.203**	**0.235**	0.194	0.174
Neural mean	0.085	0.201	0.229	**0.196**	**0.175**
LSA max	**0.099**	0.190	0.207	0.176	0.159
LSA mean	0.097	0.185	0.212	0.176	0.159
CF (baseline)	*0.093*	*0.207*	*0.236*	*0.195*	*0.175*
Content-based + collaborative filtering (CF)					
Neural max + CF	0.109	0.222	0.256	0.233	0.209
Neural mean + CF	**0.111**	0.221	0.263	**0.244**	**0.218**
LSA max + CF	0.104	**0.234**	0.259	0.235	0.210
LSA mean + CF	0.100	0.229	**0.264**	0.238	0.215

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

La contribution de ce travail est une nouvelle approche basée sur les réseaux neuronaux pour la recommandation de personne à personne dans un contexte où l'on combine une information textuelle à une approche collaborative.

Cette approche utilise des données collaboratives pour entraîner un réseau neuronal encodant chaque personne dans un espace vectoriel. Chaque individu possède le même type d'information textuelle, ce qui est dans le cas des expériences réalisées ici une série d'expertises ; celles-ci peuvent être soit une expertise possédée ou une expertise recherchée. Un réseau neuronal récurrent encode chacune des expertises. Celles-ci sont par la suite groupées par individu et réduites à un vecteur représentant cet individu. L'entraînement du réseau se fait par descente de gradient utilisant une fonction de coût en deux parties. La première partie est une minimisation de la distance entre l'encodage de deux utilisateurs multiplié par un coefficient provenant des données collaboratives. La seconde est une maximisation de la distance entre chacun de ces individus et un échantillon de tous les autres individus. L'entraînement se fait par descente de gradient en prenant comme fonction de perte la distance entre deux individus et l'inverse de la distance entre chacun de ces deux individus.

L'évaluation du modèle a été faite sur les données de la l'entreprise E-180. Il est comparé à une approche classique : l'analyse sémantique latente (*Latent Semantic Analysis* ou LSA). Les métriques de comparaisons utilisées sont la précision et le rappel (*precision* et *recall*) à différents nombres de recommandations. L'approche neuronale performe mieux au niveau du rappel, mais ne parvient pas à atteindre la précision obtenue par la LSA. L'expérience a aussi été répétée en combinant les résultats de chacune de ces approches avec une approche par filtres collaboratifs. Le modèle neuronal et la LSA ont obtenu des résultats similaires.

5.2 Limites de la solution proposée

Une limite importante de cette nouvelle approche est son application restreinte à des bases de données possédant à la fois des données collaboratives et textuelles. L'organisation à faire sur les données est aussi plus complexe que pour la LSA puisqu'il faut choisir les hyperparamètres du réseau et déterminer de quelle façon calculer les similarités entre individus à partir des données collaboratives. L'entraînement des paramètres est aussi plus difficile à faire que pour la LSA puisque celui-ci dépend en partie de l'initialisation des paramètres.

On relèvera aussi que, dans le contexte plus particulier du problème à résoudre pour E-180, les offres et requêtes sont prises en compte de la même façon lors de la recommandation. Ceci pourrait occasionner des recommandations de requête à requête ou d'offre à offre. Il serait toutefois relativement simple de pallier à ce problème en filtrant les usagers recommandés sur la base de la présence ou pas du type d'expertise voulu pour une requête ou une offre. Néanmoins, le fait que l'algorithme introduit dans ce mémoire n'utilise pas cette information additionnelle et performe malgré tout au moins aussi bien que l'approche à laquelle il est comparé, suggère que le problème est important.

5.3 Améliorations futures

Les suites à donner à ces travaux sont en deux parties. La première est l'amélioration du modèle par l'utilisation de données plus diversifiées telles que les données personnelles de chaque individu, ses données géographique et ses données d'utilisation du site. Le modèle lui-même pourrait être amélioré en utilisant une attention sur les expertises plutôt que de les combiner également.

La seconde avenue de recherche à explorer sera de mieux comprendre les prédictions faites. Le manque de précision lorsque peu de recommandations sont faites provient peut-être de la rareté des interactions. Plutôt que d'évaluer seulement sur les données historiques, un test pourrait être fait avec des utilisateurs en temps réels, les résultats duquel serait un meilleur indicateur de la justesse des recommandations.

Il serait aussi intéressant de voir l'impact de la considération du type d'expertise dans les recommandations faites pour voir si les utilisateurs cherchent quelqu'un à qui enseigner ou de qui apprendre, ou s'il cherchent simplement quelqu'un partageant des intérêts communs.

RÉFÉRENCES

C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016.

J. Akehurst, I. Koprinska, K. Yacef, L. A. S. Pizzato, J. Kay, et T. Rej, "Ccr-a content-collaborative reciprocal recommender for online dating." dans *IJCAI*, 2011, pp. 2199–2204.

T. Alashkar, S. Jiang, S. Wang, et Y. Fu, "Examples-rules guided deep neural network for makeup recommendation." dans *AAAI*, 2017, pp. 941–947.

A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, et E. H. Chi, "Latent cross : Making use of context in recurrent recommender systems", dans *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 46–54.

R. Burke, "Hybrid recommender systems : Survey and experiments", *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, et A. Mahidadia, "Procf : Generalising probabilistic collaborative filtering for reciprocal recommendation", *Advances in Knowledge Discovery and Data Mining*, 2013.

C. Chen, P. Zhao, L. Li, J. Zhou, X. Li, et M. Qiu, "Locally connected deep learning framework for industrial-scale recommender systems", dans *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 769–770.

J. Chen, H. Zhang, X. He, L. Nie, W. Liu, et T.-S. Chua, "Attentive collaborative filtering : Multimedia recommendation with item-and component-level attention", dans *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 335–344.

H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems", dans *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 7–10.

J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et D. Sampath, "The youtube video recommendation system", dans *Proceedings of the Fourth ACM Conference on Recommender Systems*, série RecSys

'10. New York, NY, USA : ACM, 2010, pp. 293–296. DOI : 10.1145/1864708.1864770. En ligne : http://doi.acm.org/10.1145/1864708.1864770

E180. (2018) E180. En ligne : https://e180.co/

M. Elahi, F. Ricci, et N. Rubens, "A survey of active learning in collaborative filtering recommender systems", *Computer Science Review*, vol. 20, pp. 29–50, 2016.

J.-P. Fauconnier. (2016) Jean-philippe fauconnier. En ligne : http://fauconnier.github.io

J. Friedman, T. Hastie, et R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA :, 2001, vol. 1, no. 10.

Y. Gong et Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network." dans *IJCAI*, 2016, pp. 2782–2788.

H. Guo, R. Tang, Y. Ye, Z. Li, et X. He, "Deepfm : a factorization-machine based neural network for ctr prediction", *arXiv preprint arXiv :1703.04247*, 2017.

X. He, L. Liao, H. Zhang, L. Nie, X. Hu, et T.-S. Chua, "Neural collaborative filtering", dans *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

B. Hidasi, A. Karatzoglou, L. Baltrunas, et D. Tikk, "Session-based recommendations with recurrent neural networks", *arXiv preprint arXiv :1511.06939*, 2015.

B. Hidasi, M. Quadrana, A. Karatzoglou, et D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations", dans *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 241–248.

T. Hofmann, "Latent semantic models for collaborative filtering", *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.

Y. Hu, Y. Koren, et C. Volinsky, "Collaborative filtering for implicit feedback datasets", dans *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 2008, pp. 263–272.

S. S. Iyengar et M. R. Lepper, "When choice is demotivating : Can one desire too much of a good thing ?" *Journal of personality and social psychology*, vol. 79, no. 6, p. 995, 2000.

K. Järvelin et J. Kekäläinen, “Cumulated gain-based evaluation of ir techniques”, *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

Y. S. Kim, A. Krzywicki, W. Wobcke, A. Mahidadia, P. Compton, X. Cai, et M. Bain, “Hybrid techniques to address cold start problems for people to people recommendation in social networks”, dans *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2012, pp. 206–217.

Y. S. Kim, A. Mahidadia, P. Compton, A. Krzywicki, W. Wobcke, X. Cai, et M. Bain, “People-to-people recommendation using multiple compatible subgroups”, dans *Australasian Joint Conference on Artificial Intelligence*. Springer, 2012, pp. 61–72.

D. P. Kingma et J. Ba, “Adam : A method for stochastic optimization”, *CoRR*, vol. abs/1412.6980, 2014. En ligne : `http://arxiv.org/abs/1412.6980`

A. Krzywicki, W. Wobcke, X. Cai, A. Mahidadia, M. Bain, P. Compton, et Y. S. Kim, “Interaction-based collaborative filtering methods for recommendation in online dating”, dans *International Conference on Web Information Systems Engineering*. Springer, 2010, pp. 342–356.

A. Krzywicki, W. Wobcke, X. Cai, M. Bain, A. Mahidadia, P. Compton, et Y. S. Kim, “Using a critic to promote less popular candidates in a people-to-people recommender system”, dans *24th Innovative Applications of Artificial Intelligence Conference 2012*, 2012, pp. 2305–2310.

A. Krzywicki, W. Wobcke, Y. S. Kim, X. Cai, M. Bain, P. Compton, et A. Mahidadia, “Evaluation and deployment of a people-to-people recommender in online dating”, dans *26th Innovative Applications of Artificial Intelligence Conference 2014*, vol. 4, 2014, pp. 2914–2921.

D. C. Lay, S. R. Lay, et J. J. McDonald, “Linear algebra and its applications, 2015”.

A. Lefebvre-Brossard, A. Spaeth, et M. C. Desmarais, “Encoding user as more than the sum of their parts : Recurrent neural networks and word embedding for people-to-people recommendation”, dans *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. ACM, 2017, pp. 298–302.

Y. Li, T. Liu, J. Jiang, et L. Zhang, “Hashtag recommendation with topical attention-based lstm”. Coling, 2016.

J. Lian, F. Zhang, X. Xie, et G. Sun, “Cccfnet : a content-boosted collaborative filtering neural network for cross domain recommender systems”, dans *Proceedings of the 26th In-*

ternational Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee, 2017, pp. 817–818.

D. Liang, J. Altosaar, L. Charlin, et D. M. Blei, "Factorization meets the item embedding : Regularizing matrix factorization with item co-occurrence", dans *Proceedings of the 10th ACM conference on recommender systems*. ACM, 2016, pp. 59–66.

Y. Liu, S. Wang, M. S. Khan, et J. He, "A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering", *Big Data Mining and Analytics*, vol. 1, no. 3, pp. 211–221, 2018.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, et J. Dean, "Distributed representations of words and phrases and their compositionality", dans *Advances in neural information processing systems*, 2013, pp. 3111–3119.

G. A. Miller, "Wordnet : a lexical database for english", *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

J. Pennington, R. Socher, et C. D. Manning, "Glove : Global vectors for word representation", dans *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. En ligne : `http://www.aclweb.org/anthology/D14-1162`

L. Pizzato, T. Rej, T. Chung, I. Koprinska, et J. Kay, "Recon : a reciprocal recommender for online dating", dans *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 207–214.

L. Pizzato, T. Rej, J. Akehurst, I. Koprinska, K. Yacef, et J. Kay, "Recommending people to people : the nature of reciprocal recommenders with a case study in online dating", *User Modeling and User-Adapted Interaction*, vol. 23, no. 5, pp. 447–488, 2013.

B. A. Potts, H. Khosravi, C. Reidsema, A. Bakharia, M. Belonogoff, et M. Fleming, "Reciprocal peer recommendation for learning purposes", dans *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. ACM, 2018, pp. 226–235.

S. Prabhakar, G. Spanakis, et O. Zaiane, "Reciprocal recommender system for learners in massive open online courses (moocs)", dans *International Conference on Web-Based Learning*. Springer, 2017, pp. 157–167.

S. Sedhain, A. K. Menon, S. Sanner, et L. Xie, "Autorec : Autoencoders meet collaborative filtering", dans *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 111–112.

E. Smirnova et F. Vasile, “Contextual sequence modeling for recommendation with recurrent neural networks”, dans *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*. ACM, 2017, pp. 2–9.

A. Spaeth et M. C. Desmarais, “Combining collaborative filtering and text similarity for expert profile recommendations in social websites”, dans *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 2013, pp. 178–189.

F. Strub, R. Gaudel, et J. Mary, “Hybrid recommender system based on autoencoders”, dans *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 11–16.

J. Tierney, “Do you suffer from decision fatigue”, *The New York Times*, 2011.

H. Wang, N. Wang, et D.-Y. Yeung, “Collaborative deep learning for recommender systems”, dans *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

X. Wang, X. He, L. Nie, et T.-S. Chua, “Item silk road : Recommending items from information domains to social users”, dans *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 185–194.

C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, et H. Jing, “Recurrent recommender networks”, dans *Proceedings of the tenth ACM international conference on web search and data mining*. ACM, 2017, pp. 495–503.

Z. Wu et M. Palmer, “Verbs semantics and lexical selection”, dans *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994, pp. 133–138.

H. Ying, L. Chen, Y. Xiong, et J. Wu, “Collaborative deep ranking : A hybrid pair-wise recommendation algorithm with implicit feedback”, dans *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2016, pp. 555–567.

S. Zhang, L. Yao, et A. Sun, “Deep learning based recommender system : A survey and new perspectives”, *arXiv preprint arXiv :1707.07435*, 2017.

K. Zhao, X. Wang, M. Yu, et B. Gao, “User recommendations in reciprocal and bipartite social networks–an online dating case study”, *IEEE Intelligent Systems*, vol. 29, no. 2, pp. 27–35, 2014.

L. Zheng, V. Noroozi, et P. S. Yu, “Joint deep modeling of users and items using reviews for recommendation”, dans *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 425–434.